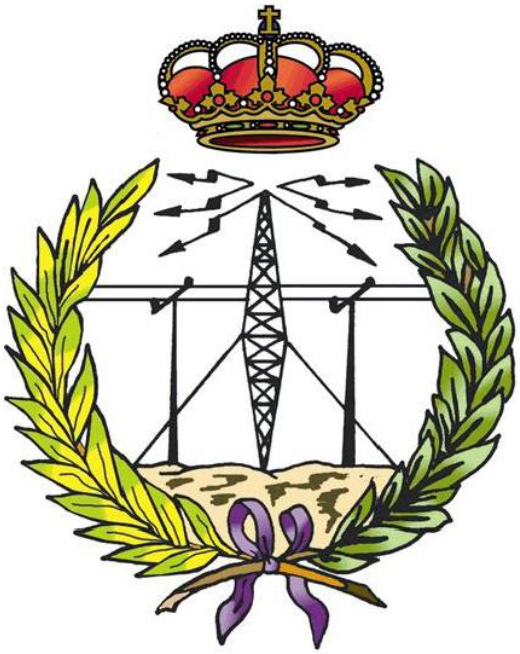
UNIVERSIDAD POLITÉCNICA DE MADRID



Escuela Universitaria de Ingeniería
Técnica en Telecomunicación

-

Proyecto Fin de Carrera

-

Sistema de control de tracción y salida para un monoplaza de la Fórmula SAE

-

José Manuel García Villegas

Septiembre 2014

E.T.S.I.S. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Aplicaciones electrónicas a la Fórmula SAE

TÍTULO: Sistema de control de tracción y salida para un monoplaza de la Fórmula SAE

AUTOR: José Manuel García Villegas

TUTOR: Pedro Cobos Arribas **Vº Bº.**

DEPARTAMENTO: SEC

Miembros del Tribunal Calificador:

PRESIDENTE: Sara Lana Serrano

VOCAL: Pedro Cobos Arribas

VOCAL SECRETARIO: José Antonio Herrera Camacho

DIRECTOR:

Fecha de lectura: 30 de Septiembre de 2014

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

Este proyecto detalla el diseño, fabricación y montaje de un sistema de control de tracción en un monoplaza que compite en la Fórmula SAE.

El control de tracción es un sistema de seguridad del automóvil diseñado para prevenir la pérdida de adherencia cuando alguna rueda presenta deslizamiento, bien porque el conductor se excede en la aceleración o bien porque el firme esté resbaladizo.

El sistema se activa cuando detecta un deslizamiento en alguna de sus ruedas motrices, para ello evalúa en todo momento la velocidad a la que giran las ruedas del vehículo y compara sus velocidades de giro. Cuando el sistema se activa, hace que el motor disminuya su potencia hasta que con este efecto conseguimos traccionar correctamente y mantener el deslizamiento en unos valores controlados.

Índice

1. Resumen

En este proyecto de final de carrera se detalla el proceso de diseño, fabricación, montaje y ajuste de un dispositivo electrónico que sirva como sistema de control de tracción de un vehículo y que acoplaremos sobre un monoplaza de carreras que participa en la competición Fórmula SAE. La Fórmula SAE (Society of Automotive Engineers - Sociedad de Ingenieros de Automoción), es una competición de coches de carreras monoplaza a nivel universitario que promueve el desarrollo de la ingeniería aplicada a la automoción.

Se pretende que este libro sirva de guía para el correcto manejo y desempeño del sistema fabricado. Además se ha pretendido que su lectura resulte fácil y comprensible para que la persona que lea este libro sea capaz de entender el sistema realizado para así poderlo mejorar.

Gracias a la colaboración entre la Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST) de la Universidad Politécnica de Madrid (UPM), la Escuela de Ingenieros Industriales de esta misma Universidad (ETSII) y el Instituto Universitario de Investigación del Automóvil (INSIA), se sientan las bases de una plataforma docente en la cual se posibilita la formación y desarrollo de un vehículo tipo fórmula que participa en la ya mencionada competición Fórmula SAE.

Para ello, se formó en el 2003 el equipo UPMRacing, primer representante español en el evento. El equipo se compone de más de 50 alumnos de la UPM y del Máster de Ingeniería en Automoción del INSIA. Es por tanto, en el vehículo fabricado por el equipo UPMRacing, en el que se pretende instalar este sistema de control de tracción.

El control de tracción es un sistema de seguridad del automóvil diseñado para prevenir la pérdida de adherencia cuando alguna rueda presenta deslizamiento, bien porque el conductor se excede en la aceleración o bien porque el firme esté resbaladizo.

La unidad de procesamiento del sistema de control de tracción fabricado lee la velocidad de cada rueda del vehículo mediante unos sensores y determina si existe deslizamiento, en tal caso, manda una señal a la centralita para disminuir la potencia hasta que el deslizamiento disminuya a unos valores controlados. El sistema cuenta con un control remoto que sirve como interfaz para que el piloto pueda manejarlo. Por último, el dispositivo es capaz de conectarse a un bus de comunicaciones CAN para configurar ciertos parámetros.

El objetivo del sistema es, básicamente, hacer que el coche no derrape en aceleraciones fuertes; concretamente en las salidas desde parado y al tomar una curva, aumentando así la velocidad en circuito y la seguridad del piloto.

2. Abstract

The purpose of this project is to describe the design, manufacture, assembly and adjustment processes of an electronic device acting as the traction control system (TCS) of a vehicle, that we will attach to a single-seater competition formula SAE car.

The Formula SAE (Society of Automotive Engineers) is a graduate-level single-seater racing car competition promoting the development of automotive applied engineering.

We also intend this work to serve as a technical user guide of the manufactured system. It is drafted clearly and concisely so that it will be easy for all those to whom it is addressed to understand and subject to further improvements.

The close partnership among the Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación (ETSIST), Escuela de Ingenieros Industriales (ETSII) of Universidad Politécnica de Madrid (UPM), and the Instituto Universitario de Investigación del Automóvil (INSIA), lays the foundation of a teaching platform enabling the training and development of a single-seater racing car taking part in the already mentioned Formula SAE competition.

In this respect, UPMRacing team was created back in 2003, first spanish representative in this event. The team consists of more than 50 students of the UPM and of INSIA Master in Automotive Engineering. It is precisely the vehicle manufactured by UPMRacing team where we intend to install our TCS.

TCS is an automotive safety system designed to prevent loss of traction when one wheel has slip, either because the driver exceeds the acceleration or because the firm is slippery.

The device's central processing unit is able to detect the speed of each wheel of the vehicle via special sensors and to determine wheel slip. If this is the case, the system sends a signal to the ECU of the vehicle to reduce the power until the slip is also diminished to controlled values. The device has a remote control that serves as an interface for the pilot to handle it. Lastly, the device is able to connect to a communication bus system CAN to set up certain parameters.

The system objective is to prevent skidding under strong acceleration conditions: standing-start from the starting grid or driving into a curve, increasing the speed in circuit and pilot's safety.

3. Fórmula SAE

3.1. ¿Qué es?

La Fórmula SAE, es una competición de coches de carreras monoplaza a nivel universitario que promueve el desarrollo de la ingeniería aplicada a la automoción. Los participantes a esta competición son equipos que representan a universidades de países de todo el mundo. El nombre viene dado por la SAE, Society of Automotive Engineers (Sociedad de Ingenieros de Automoción) ahora llamada SAE International, la cual organiza los eventos.

La SAE International se creó en EE. UU. en 1905. El principal objetivo era el desarrollo de los estándares para todos los tipos de vehículos, incluyendo coches, camiones, barcos, aviones, etc. Hoy en día, según datos de su página web www.sae.org, la organización se compone de 115.000 miembros repartidos en más de 100 paises en todo el mundo.

La SAE Internatinal organiza doce competiciones de diseño entre universitarios llamadas Collegiate Design Competitions. Estos eventos retan a los estudiantes a diseñar y construir vehículos funcionales en un ámbito competitivo. Estas competiciones de diseño reúnen a más de 4.500 estudiantes de 500 universidades de todo el mundo. Entre las competiciones organizadas están la Formula SAE, Formula Hybrid, SAE Aero Design, SAE Mini Baja y SAE Clean Snowmobile Challenge.

3.2. Historia

La Fórmula SAE tiene su origen en EE. UU. La primera competición empezó a gestarse en 1979 cuando Mark Marshek, docente de la Universidad de Houston,

contactara con el Departamento de Relaciones Educativas de la SAE un año antes. El concepto original de la Fórmula SAE era una evolución de la Baja SAE, una competición existente en la que el tipo de vehículo a construir por los estudiantes es similar a un car-cross (Figura 1). Sin embargo, esta competición limitaba mucho la libertad (motor proporcionado por la organización sin posibilidad de modificarlo) y la nueva competición debía darles mayor margen para diseñar el monoplaza.

Figura 1: Vehículo participante en la competición Baja SAE

Así se llega a 1981, año en que se organiza, en la Universidad de Tejas (Austin), la primera edición de la Formula SAE. Participaron 6 equipos y un total de 40 alumnos.

Esta competición fue creciendo hasta que en 1998 se celebró la primera edición del evento fuera de EE. UU. En este año, dos monoplazas Estadounidenses y dos Ingleses compitieron en una demostración en el área de pruebas de la MIRA (Motor Industry Research Association), en la ciudad de Nuneaton, condado de Warwickshire, Inglaterra. La iniciativa fue considerada muy útil para el desarrollo del aprendizaje y habilidades prácticas de los estudiantes. Así, la "Institution of Mechanical Engineers" de Inglaterra, aceptó la gestión de este evento Europeo en una asociación con la SAE. Este evento fue llamado Formula Student. La Formula Student es ligeramente diferente de la Fórmula SAE, ya que está diseñada para ser un ejercicio de aprendizaje progresivo a lo largo de un curso de tres o cuatro años académicos. Sin embargo, se utilizan las mismas reglas para la Formula Student y la Fórmula SAE (con algunos cambios de menor importancia) y esto significa que los equipos de los estudiantes pueden participar con sus coches en ambos eventos.

Posteriormente, se crearon eventos en otros paises como Alemania, Japón, Brasil, Australia, etc. Todas ellas utilizan la misma normativa base original de la Formula SAE y llegan a albergar hasta 120 equipos y más de 2000 estudiantes.

Los equipos de las universidades participan en alguna de las competiciones que se celebran por todo el mundo. Los resultados de las competiciones son recogidos y puntúan en un ranking mundial.

3.3. Objetivo de la competición

El reto de los equipos es desarrollar un vehículo que sea capaz de superar con éxito cada una de las pruebas de la competición y ganará el que mayor puntuación consiga en base a esas pruebas. Los mejores equipos entran en un ranking mundial.

Para el propósito de la competición, los equipos han de asumir el papel de trabajadores de una firma automovilística, que diseña, fabrica, testea y realiza una demostración de un prototipo de un monoplaza que quieren vender a una hipotética corporación que está considerando la producción de un vehículo de competición.

Todo esto conforma una inestimable base para el desarrollo de las habilidades y capacidades de los estudiantes, puesto que se pretende que el futuro ingeniero siga todas las fases de la vida de un producto.

Para juzgar la excelencia del trabajo realizado por cada equipo, se realizan una serie de pruebas, las cuales se puntúan como muestra la Tabla 1. Más adelante se explicará detalladamente en qué consiste cada una.

Figura 2: Eslogan de la Fórmula SAE

Pruebas Estáticas:	
Inspección Técnica	Sin puntuación
Presentación del Proyecto	75
Diseño	150
Análisis de costes	100
Total	325
Pruebas Dinámicas:	
Aceleración	75
Skid-Pad	50
Autocross	150
Resistencia	300
Economía de combustible	100
Total	675
Total de las pruebas	1000 puntos

Tabla 1: Información sobre las pruebas y su puntuación.

3.3.1. Las categorías

En las competiciones de Formula Student existen 2 categorías, o clases. El mayor número de equipos se presenta a la categoría primera (Clase 1), que es aquella en la que compiten los monoplazas acabados y que hayan participado por primera vez hace como máximo un año. Pero también existe una categoría para que se presenten equipos que están comenzando y no tienen el vehículo completamente preparado.

Clase 1 En esta categoría participan monoplazas totalmente construidos y que son capaces de moverse. Es la categoría más importante de todas y un mismo coche solamente puede participar en ella durante los 12 meses siguientes a la primera

competición en la que participa. Esta regla obliga a los equipos a progresar y fabricar cada año nuevos vehículos. Se permite reaprovechar los componentes de monoplazas antiguos, salvo el chasis. Los equipos de esta categoría puntúan en todas las pruebas: las estáticas y las dinámicas. A partir de 2012, también los vehículos eléctricos o con combustibles alternativos participan en esta categoría (anteriormente lo hacían en una denominada: Clase 1A).

Clase 2 Se trata de la clase de entrada para los equipos nuevos o estudiantes noveles de un equipo que también compita en la Clase 1 y que estén en fase de aprendizaje y desarrollo de un monoplaza. En esta categoría, participan únicamente vehículos en fase de diseño y validación del modelo, aunque si el equipo lo desea, puede incluir piezas fabricadas y sistemas que se hayan producido. Como norma general y para fomentar el progreso, un equipo no se puede presentar a esta categoría dos años consecutivos, teniendo que hacerlo en la superior. Únicamente se puntúan las pruebas de diseño, presentación y coste.

3.3.2. Las pruebas

Los requisitos respecto al vehículo y a los participantes, así como todo lo relacionado con las pruebas y reglas del evento está recogido en un documento que la SAE actualiza cada temporada. En las siguientes líneas nos centraremos en explicar brevemente en qué consisten las pruebas que deben pasar los equipos. Para mayor detalle se debe consultar el documento oficial en la web de la SAE International, el cual se actualiza cada temporada.

Las pruebas se engloban dentro de dos tipos: las pruebas estáticas y las pruebas dinámicas.

Pruebas estáticas Estas pruebas se realizan con el coche parado y apagado. La suma de puntos de este tipo de pruebas es de 325 como máximo. Se realizan cuatro pruebas pero una de ellas, la prueba de inspección técnica, no es puntuable.

1. **Inspección técnica:** El objetivo de esta prueba es determinar si el vehículo cumple con las reglas de la FSAE, en cuanto a requerimientos y restricciones. El vehículo debe pasar todas las partes de la inspección técnica antes de permitírsele participar en cualquier prueba dinámica o correr en el circuito de pruebas. Las partes en las que se divide la inspección técnica son: inspección de seguridad, prueba de inclinación, prueba de ruido y prueba de frenado.

2. **Presentación del proyecto:** El objetivo de esta prueba es evaluar la capacidad de cada equipo para vender el producto a los responsables de la hipotética corporación que quiere fabricar un vehículo de carreras. En la presentación se debe tratar de convencer a estos responsables de la superioridad del diseño frente al de otros competidores. Los jueces evalúan tanto la organización como el contenido y otorgan una puntuación en base a esto. La máxima puntuación que se otorga en esta prueba es de 75 puntos.

3. **Análisis de costes:** El objetivo de esta prueba es aleccionar a los participantes de que el coste y el presupuesto son factores críticos y que deben tenerse en cuenta en cualquier proceso de ingeniería. Se espera que los miembros del equipo aprendan el concepto de "trade-off" y lleguen a una solución de compromiso entre diferentes partes del proyecto, sopesando cuándo hay que sacrificar en ciertas ventajas de menor valor para ganar en otras más valiosas. La puntuación máxima alcanzable en esta prueba es de 100 puntos. Su evaluación se realiza en base a dos hitos:

   - La elaboración y presentación de un informe escrito que debe ser enviado a los jueces antes de la competición.
   - La defensa del coste de cualquiera de las partes del vehículo en un evento que se realiza durante los días de competición.

4. **Diseño** El objetivo es evaluar el trabajo de ingeniería en el diseño del coche y cómo éste cumple con las exigencias del mercado. Así, durante esta prueba, el diseño de los componentes del vehículo es puesto en tela de juicio por ingenieros y diseñadores de la industria del automóvil. Hay que mostrar todos los elementos del coche y defender las diferentes soluciones técnicas adoptadas. Así mismo, se requiere que los participantes den justificaciones a las soluciones escogidas y muestren los resultados que los diferentes test arrojaron sobre las mismas, con el fin de verificar la validez del diseño. La puntuación máxima que se puede obtener en esta prueba es de 150 puntos.

Pruebas dinámicas Como su propio nombre indica, en estas pruebas se evalúan las capacidades dinámicas del vehículo, por lo que se demostrarán sus aptitudes en 5 diferentes aspectos. Hay que resaltar que por la naturaleza de las pruebas, el piloto o pilotos implicados en ellas, juegan un papel importante en el desempeño del monoplaza. Para participar en este bloque de pruebas es necesario pasar con éxito la inspección técnica. La suma de puntos de este bloque de pruebas es de 675 como máximo, que sumados a los 325 puntos de las pruebas estáticas nos da que la puntuación máxima repartida entre todas las pruebas es de 1000 puntos.

1. **Aceleración:** Se mide la aceleración del monoplaza en línea recta, partiendo desde parado sobre una superficie plana. El coche deberá recorrer 75 *metros*. La parte más exterior de la delantera del coche deberá estar 0,30 *metros* por detrás de la línea de salida. Se comenzará a contar el tiempo una vez que el vehículo cruce la marca de salida. Cada equipo podrá realizar dos tandas y en cada una de ellas se podrán realizar dos intentos. Cada tanda la realizará un conductor distinto. El mejor tiempo logrado en todos los intentos es el que computa de cara a la puntuación obtenida. La máxima puntuación que se puede obtener en esta prueba es de 75 puntos.

2. **Skid-Pad:** El objetivo de la prueba es medir la habilidad de giro del vehículo en una superficie plana mientras efectúa un giro de radio constante. Cada coche

puede realizar dos tandas, con un piloto distinto en cada tanda y con dos intentos en cada una de ellas. El circuito consta de dos círculos de 15,25 m de diámetro dispuestos en forma de ocho. Los centros de los círculos estarán separados 18,25 m, teniéndose un carril con una anchura de 3 m que está marcado por conos de distintos colores a cada lado. La línea de comienzo y de parada estará determinada por la línea que une los centros de los dos círculos. Los coches entran en perpendicular al ocho y deberán dar dos vueltas completas al círculo derecho y seguidamente se entrará en el izquierdo y se completarán otras dos vueltas. El esquema del circuito se puede ver en la Figura 3. Por cada cono que se derribe se tendrá una penalización de 0,25 *segundos*. El tiempo que se computa en cada uno de los intentos es el correspondiente al segundo giro completado a derechas más el segundo giro completado a izquierdas. La máxima puntuación que se puede obtener en esta prueba es de 50 puntos.

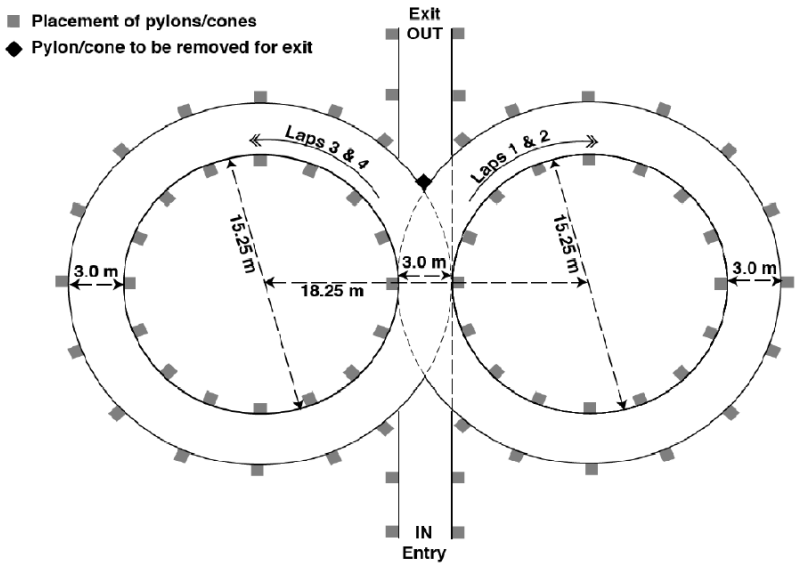


Figura 3: Vista en planta del diseño del circuito del Skid-Pad

3. **Autocross:** El objetivo de esta prueba es evaluar la manejabilidad del vehículo en un circuito muy virado y en ausencia de otros vehículos competidores. Esta prueba combina la capacidad del coche para acelerar, frenar y tomar curvas muy cerradas. Al igual que en las pruebas anteriores se tiene la posibilidad de realizar dos tandas. De las cuatro vueltas que den los dos pilotos se contabiliza como

tiempo de la prueba el mejor de los cuatro. La velocidad media que se alcanzará en esta prueba está entre 40-48 *km/h*. Las rectas no son de una longitud mayor a 60 *m*. Los radios de giro van de 23 a 45 *m*. Las zonas de slalom estarán formadas por conos separados una distancia que puede variar entre 7,62 *m* y 12,19 *m*. La longitud de cada vuelta al circuito es de aproximadamente 0,805 *km*. Cada cono derribado durante la carrera se penaliza con dos segundos sobre el tiempo total. La puntuación máxima que se puede obtener en este evento es de 150 puntos.

4. **Resistencia y economía de combustible:** Esta prueba está diseñada para evaluar el rendimiento general del coche y así testear su durabilidad y fiabilidad. La economía de combustible del monoplaza se mide en conjunto con la prueba de resistencia. Esta prueba se realiza en una sola tanda que consta de unos 22 *km* dentro de los cuales no se puede reparar el vehículo. Para evaluar el consumo lo que se hace es llenar completamente el depósito de los vehículos al comienzo de la prueba y una vez que el vehículo completa con éxito todas las vueltas al circuito, se vuelve a rellenar, midiéndose así cuanto ha sido el consumo total durante la prueba. De esto se deduce que no está permitido el repostaje durante la prueba. La velocidad media que se alcanza en esta prueba está comprendida entre 48 y 57 *km/h*, pudiendo alcanzar máximas de unos 105 *km/h*. Las rectas no suelen ser mayores de 77 m. Los radios de giro están comprendidos entre 30 y 54 *m*. En la zona de slalom los conos tienen una separación de entre 9 y 15 *m*. A mitad de carrera se realizará un cambio de piloto que deberá durar como máximo tres minutos. La puntuación máxima que se obtiene en esta prueba de resistencia es de 300 puntos según el tiempo empleado en recorrer los 22 *km* en el circuito y de 100 puntos adicionales que se otorgan en función del más bajo consumo de combustible.

3.4. Fórmula SAE en España

En España se va incrementando el número de equipos participantes en la Fórmula SAE. El equipo español pionero en participar en este tipo de competiciones fue el UPMRacing creado en 2003. Hasta 2006 no se creó el segundo equipo español, el FSBizkaia. La representación española está creciendo año tras año. Además, se ha consolidado la edición española de la Fórmula Student en Cataluña que en el año 2010 tuvo su primer evento (del 23 al 26 de Septiembre de 2010) y que ha continuado en los años sucesivos.

Actualmente, después de la competición de Michigan (17-Mayo-2014) 8 equipos españoles se encuentran en el ranking mundial de la FSAE para coches de combustión, tal y como muestra la Tabla 2.

Ranking	Equipo
176	ETSEIB Motorsport
235	FSBizkaia
291	Tecnun Motorsport
324	UPMRacing
369	FórmulaUEM
408	UPCT Racing Team
439	ARUS Andalucia Racing
470	UCLM Racing Team

Tabla 2: Ranking mundial FSAE 17-Mayo-2014 de coches de combustión.

En cuanto a coches con motor eléctrico, después de la competición de Australasia (15-Diciembre-2013), 4 equipos se encuentran en el ranking mundial de la FSAE, como muestra la Tabla 3

Ranking	Equipo
37	ETSEIB E-Motorsport
40	FSBizkaia
59	UPM Electric Racing
61	Tecnun Seed Group

Tabla 3: Ranking mundial FSAE 15-Diciembre-2013 de coches eléctricos.

3.4.1. Equipos españoles

Algunos de los equipos españoles son los siguientes:

UPMRacing Es el equipo de la Universidad Politécnica de Madrid. Fue el primer equipo español, creado en Noviembre de 2003 por iniciativa del Instituto Universitario de Investigación del Automóvil (INSIA) y de la Escuela Técnica Superior de Ingenieros Industriales de la Universidad Politécnica de Madrid (UPM) y contando con el apoyo del Máster en Ingeniería en Automoción del INSIA.

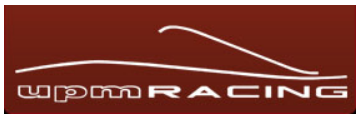


Figura 4: Logo del equipo UPMRacing

El equipo se compone de más de 50 alumnos de la Escuela Técnica Superior de Ingenieros Industriales, del Máster de Ingeniería en Automoción del INSIA, de la Escuela de Ingeniería Técnica Aeronáutica y de Escuela Universitaria de Ingenieros Técnicos en Telecomunicaciones.

El equipo se encuentra estructurado en 7 divisiones: Aerodinámica, Chasis, Dinámica Vehicular, Frenos, Electrónica, Motor y Marketing. Cada división está liderada por un responsable que se encarga de coordinar y gestionar el trabajo de

Figura 5: Equipo UPMRacing

Figura 6: Monoplaza de la temporada 2009-2010: el UPM007.

todos sus miembros, tomar decisiones técnicas y actuar como portavoz en las reuniones técnicas generales.

Tienen un equipo para competir con un monoplaza eléctrico, el UPM Electric Racing.

ETSEIB Motorsport Es el equipo de la Universidad Politécnica de Cataluña y tiene su sede en la Escola Tècnica Superior d'Enginyeria Industrial de Barcelona. Fue fundado en la temporada 07/08.

Figura 7: Logo del equipo ETSEIB Motorsport

En la temporada 2010-2011 el equipo consiguió ser por primera vez en su historia el mejor equipo español, alcanzando la posición 107 después de la competición italiana (6-Septiembre-2010). En el evento organizado en España en 2010, en el Circuito de Montmeló, logró la quinta posición.

También tienen el equipo ETSEIB E-Motorsport para competir con monoplaza eléctrico.

TECNUN Motorsport Es el equipo de la Universidad de Navarra, con sede en el campus tecnológico de TECNUN (San Sebastián, Guipúzcoa). El equipo de TECNUN Motorsport, está integrado por estudiantes de Ingeniería Industrial e Ingeniería de Telecomunicaciones.

Figura 8: Logo del equipo TECNUN Motorsport

Su debut fue en el año 2009, en el circuito de Silverstone donde el equipo entró por primera vez a formar parte de esta prestigiosa competición. Tras su sorprendente debut, participó también en la Formula Student Alemana, culminando de esta forma la primera temporada del equipo TECNUN Motorsport.

Han creado también un equipo para competir con un monoplaza eléctrico, el Tecnun Seed Group.

FSBizkaia Lo conforman un grupo de estudiantes de la Escuela Técnica Superior de Ingeniería de Bilbao. Para este reto cuentan con el inestimable apoyo del Departamento de Ingeniería Mecánica de la UPV, diferentes empresas e instituciones y varios centros formativos.

Figura 9: Logo del equipo FSBizkaia

El equipo FSBizkaia comenzó en 2006 como un proyecto innovador del Departamento de Ingeniería Mecánica de la Escuela Técnica Superior de Ingeniería de Bilbao. Tras dos años de preparación e investigación, en la temporada 2007-2008 llegó el primer prototipo y en el siguiente año el segundo monoplaza, con grandes mejoras en su diseño.

En la temporada 2009-2010 se inició, también, las investigaciones necesarias para la construcción de un prototipo con sistema de propulsión eléctrico y el equipo ya se encuentra en el ranking mundial de la FSAE tanto en la modalidad de monoplaza de combustión como eléctrico.

Fórmula UEM Es el equipo de la Universidad Europea de Madrid. Comenzó en el año 2009 a participar en las competiciones de Formula Student que se realizan en Silverstone.

UPC ecoRacing Es un equipo de trabajo formado por estudiantes de Ingeniería Industrial y Aeronáutica de la Escuela Técnica Superior de Ingenierías Industrial y Aeronáutica de Tarrasa(ETSEIAT), cuyo principal objetivo es diseñar y fabricar un coche de competición híbrido que aúne altas prestaciones y la máxima eficiencia energética en el marco de la Formula Student.

Figura 10: Logo del equipo UPC ecoRacing

ARUS Andalucia Racing Es el equipo de la Universidad de Sevilla. Fue creado en 2012 y durante el primer año, el equipo ha dedicado todos sus esfuerzos en sentar unas bases sólidas para organizar un proyecto de esta magnitud, centrándose en el diseño y el aprendizaje.

La primera aparición oficial de ARUS fue en septiembre de 2013 en Formula Student Spain (FSS), celebrado en el Circuit de Catalunya, donde participaron en las pruebas estáticas. En el evento de 2014 en la Formula Student Spain, celebrado en Montmeló, en el Circuit de Catalunya, ha conseguido terminar con éxito todas las pruebas, incluida la difícil prueba del Endurance.

UCLM Racing Team Es el equipo de la Universidad de Castilla-La Mancha, equipo que se creó en el 2012. Durante el curso 2012-13 se desarrollo el primer prototipo FSURT-01 con el que participaron en la competición de Varano di Melegari en Italia.

EUETIB e-Tech Racing Es un equipo de Formula Student formado por estudiantes de la Escuela Universitaria de Ingeniería Técnica de Barcelona (EUETIB), centro adscrito a la Universidad Politécnica de Cataluña. El equipo fue fundado en el curso 2012-2013, en el cual empezó con el diseño de un monoplaza eléctrico.

UPCT Racing Team Es el equipo formado por los estudiantes de la Universidad Politécnica de Cartagena.

4. Estado del arte

4.1. Introducción

Desde el inicio de la creación de los automóviles, éstos se han encontrado con diversos aspectos que ponían en riesgo la vida del conductor y sus ocupantes. Estos factores de riesgo se han ido subsanando a lo largo de la historia con la implantación de sistemas de seguridad en el automóvil, que han llegado hoy en día a lograr un alto grado de protección y seguridad cuando montamos en un coche.

Entre los elementos más destacados en seguridad automovilística podemos citar los cinturones de seguridad, los reposacabezas, el Airbag, el ABS, el control de tracción y el ESP.

Algunos de estos sistemas, además de ofrecer mayor seguridad al conductor, también hacen más eficiente el desempeño del vehículo en determinadas circunstancias. Por ejemplo, el uso del sistema antibloqueo de frenos (ABS), nos proporciona mayor seguridad en la frenada disminuyendo la posibilidad de que el coche pierda el control y además, en la mayoría de los casos, repercute en una menor distancia de frenado. Por todo ello, se ve totalmente necesario que se continúe con una labor de investigación por parte de las entidades involucradas en la fabricación y desarrollo de automóviles.

Dicho lo cual, se tratará ahora sobre uno de los subsistemas más conocidos dentro del mundo del automóvil: el control de tracción. Para hablar sobre el control de tracción, primero se va a tratar la causa que hace que se necesiten estos sistemas, por ello se explicarán elementos de la dinámica de un coche y se evidenciarán los momentos en los que existe una carencia que debemos subsanar para un mejor funcionamiento dinámico del vehículo.

4.2. Dinámica en el automovil: Diferenciales

Todos tenemos claro que cuando llueve hay que tener más cuidado en la conducción porque la adherencia de los neumáticos con el suelo es mucho menor y una fuerte aceleración o un giro brusco puede provocar una situación incontrolable que finalmente se trasforme en un accidente.

Un elemento básico e indispensable de cualquier vehículo móvil son las ruedas. Las ruedas, con sus neumáticos, son los elementos que nos unen físicamente a la superficie por la cual nos queremos desplazar. Son a dichas ruedas a las que debemos transferir la fuerza que genera el motor y van a determinar la dinámica y la motricidad del vehículo.

Cuando se construyeron los primeros coches, los ingenieros se encontraron con un problema ligado al movimiento de las ruedas. En los primeros diseños se disponía de un eje que era solidario al movimiento de las ruedas que se encontraban a sus extremos (ver Figura 11). Esto provoca que cuando gira el eje, giran por igual las ruedas a sus extremos. Cuando el coche marcha en línea recta, las ruedas van a la misma velocidad y el sistema anteriormente mencionado funciona, pero cuando el coche toma una curva, las ruedas interiores recorren menos distancia que las exteriores en un mismo tiempo y el anterior sistema no deja que esto ocurra (Figura 12). Esto implica que el vehículo tienda a ir en línea recta aún tomando una curva y hace que la rueda exterior tenga una tendencia a bloquearse y la interior a "sobregirar". Todo esto causa un comportamiento inestable en el coche y provoca patinamientos en las ruedas ya que no pueden describir con exactitud el giro.

La solución al anterior problema estaba en la inclusión del mecanismo diferencial (Figura 13). Este mecanismo se acopla al eje y permite que las ruedas del mismo eje giren a distintas velocidades en una curva, incluso una de ellas puede estar parada. Nótese que la inclusión del diferencial hace que el eje quede dividido en dos semiejes o palieres.

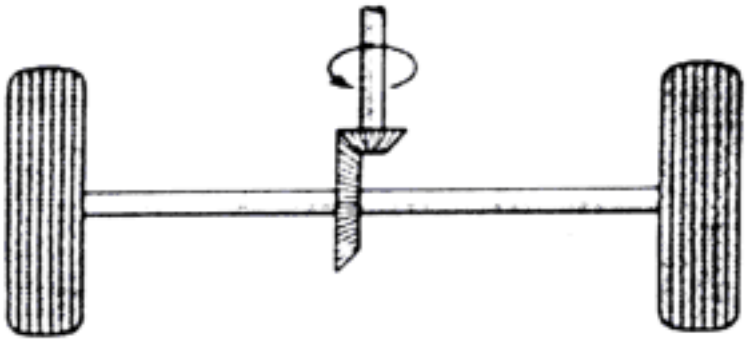


Figura 11: Eje solidario a las ruedas.

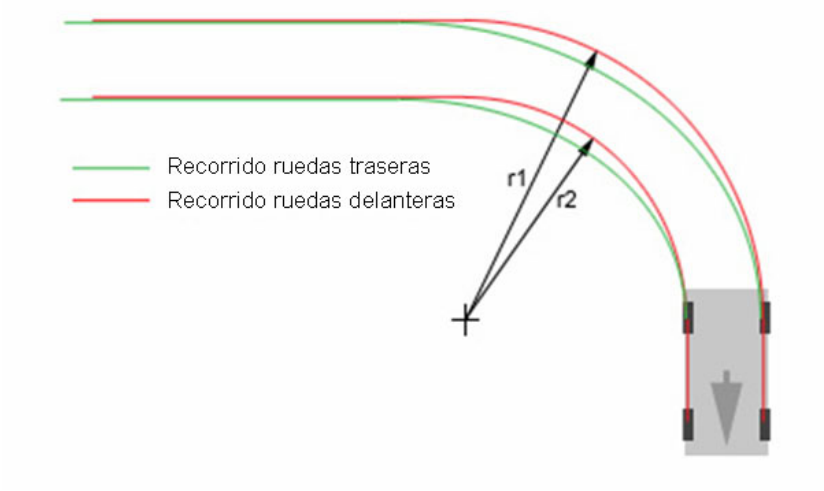


Figura 12: Recorrido de las ruedas en una curva.

Algunas de las razones por las que se acopla un diferencial es que se absorbe menos potencia del motor para girar, se consigue un menor radio de giro del vehículo para un determinado ángulo de giro de las ruedas y permite realizar maniobras a baja velocidad con precisión.

4.2.1. Partes y funcionamiento

El diferencial consta de unos engranajes y piñones acoplados dentro de una carcasa. En la Figura 14 podemos observar sus partes. Existe una corona que recibe el movimiento a través del elemento denominado árbol de transmisión. Vemos que de

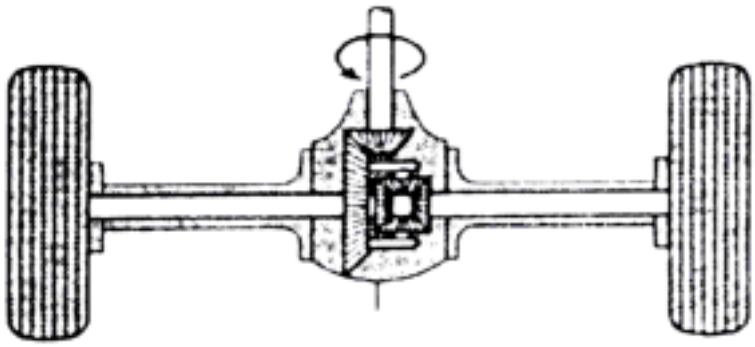


Figura 13: Eje con un diferencial acoplado.

la corona salen dos brazos que están unidos a ésta. En el extremo de estos brazos se colocan unos engranajes llamados satélites que van montados sobre un pequeño eje que les permite girar sobre si mismos. Por último ser ven otros engranajes llamados planetarios que son solidarios al semieje o palier. Nótese que en la imagen, el movimiento del motor se transmite a través del árbol de trasmisión. En otras configuraciones, como por ejemplo vehículos de tracción delantera, el movimiento se puede trasmitir ala corona a través de otro elemento, como un engranaje.

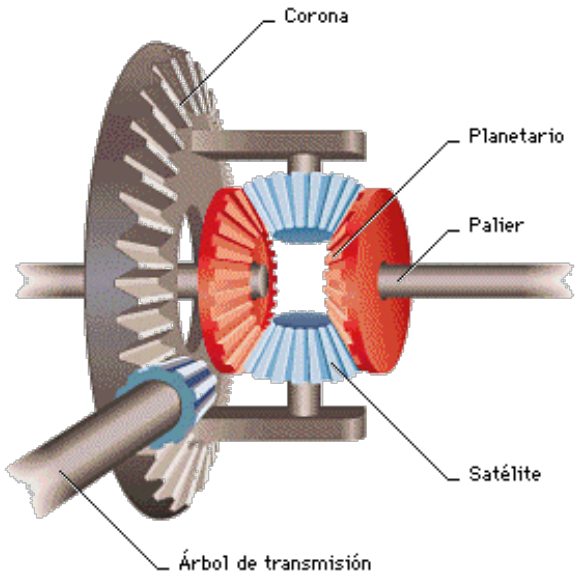


Figura 14: Partes de un diferencial.

El diferencial hace que el reparto de par entre los dos semiejes sea el mismo, es decir, hace que la fuerza generada por el motor, y que se transfiere a través del árbol

de transmisión, se divida por igual a cada rueda. No debemos confundir el par con la velocidad de giro. El par es una fuerza aplicada a la rueda para hacerla girar. En una rueda que presente baja resistencia al movimiento, por ejemplo una rueda al aire, podemos lograr que gire rápidamente aplicando un par reducido, sin embargo, en una rueda frenada no conseguiremos que se mueva aunque apliquemos mucho par.

El funcionamiento del diferencial es muy interesante:

Al girar el árbol de transmisión por acción del motor, se hace girar la corona. Como se ve en la Figura 15, al girar la corona se mueven los satélites ya que están acoplados a los brazos que salen de la corona. Los satélites describirán un movimiento circular en un plano perdendicular al eje. Si existe suficiente adherencia de las ruedas de ambos semiejes o palieres, los satélites "arrastrarán" consigo a los dos planetarios haciéndolos girar por igual y dado que los planetarios son solidarios con los palieres, éstos también comenzarán a girar, proporcionando así el movimiento de las ruedas.

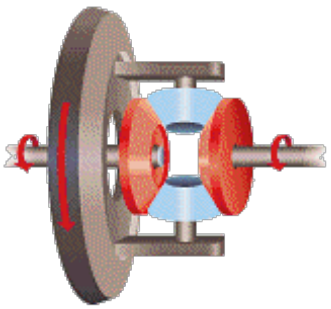


Figura 15: Marcha en línea recta.

Imaginemos por un momento que subimos el coche a una plataforma dejando así las ruedas al aire. Ahora una de las ruedas del eje la vamos a frenar fuertemente para que no gire, así su palier y por tanto su planetario tampoco se podrán mover. Ahora comenzamos a girar el árbol de transmisión, haciendo así que gire la corona con sus brazos y también los satélites. Visionemos por un momento que el planetario anteriormente referenciado no se puede mover lo que provocará que automáticamente los satélites comiencen a girar sobre si mismos. Al girar los satélites sobre si mismos permiten que este palier del que hablamos pueda quedar parado sin problemas, pero

el planetario del otro palier sí que comenzaría su movimiento "arrastrado" por los satélites.

En una curva el diferencial hará que el giro de las ruedas sea ni más ni menos que el apropiado, en contraposición con el sistema del eje solidario a las ruedas. Podemos ver en la Figura 16 lo que ocurre en el diferencial cuando afronta un giro.

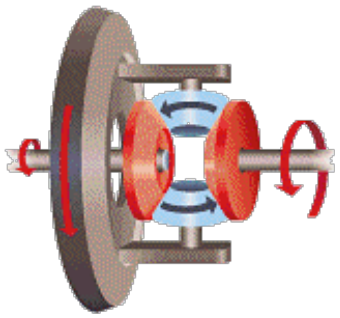


Figura 16: Diferencial afrontando un giro.

El sistema de diferencial que acabamos de explicar se conoce como diferencial libre y hemos conocido sus ventajas, pero también tiene algún inconveniente: si por ejemplo se queda una de las ruedas motrices del vehículo al aire o atrapada en un banco de arena, al acelerar sólo conseguiremos generar movimiento en esa rueda, ya que es la que menos oposición al movimiento presenta y no lograremos hacer avanzar al coche.

4.2.2. Diferenciales de deslizamiento limitado

Siguiendo con el tema de los diferenciales, es conveniente saber que lo contrario al diferencial libre es el diferencial bloqueado. Un diferencial bloqueado hace que los semiejes o palieres se hagan solidarios entre si, comportándose por tanto como un único eje. Es bien fácil conseguir esto: lo único que tenemos que hacer es soldar los satélites a su eje para evitar que puedan girar sobre si mismos.

Bien, para solventar el anterior inconveniente de los diferenciales libres, se han desarrollado diferenciales que están a medio camino entre el diferencial libre y

el bloqueado permanente. Los hay de tantos tipos como situaciones nos podemos encontrar. Los hay que bajo poca diferencia de resistencia (oposición al movimiento) entre las ruedas, hacen que los semiejes o palieres sean solidarios; otros, en cambio, comienzan siendo libres bajo una diferencia de velocidad entre semiejes relativamente pequeña y se van bloqueando progresivamente; otros permanecen libres si no existe potencia por parte del árbol de transmisión y se auto-bloquean cuando ésta está presente; etc.

Habitualmente estos diferenciales se suelen conocer como diferenciales de deslizamiento limitado (LSD por sus siglas en inglés) o diferenciales autoblocantes.

No debemos olvidar que los diferenciales también se acoplan a vehículos con tracción a las 4 ruedas. En estos vehículos, aunque no en todos, se coloca además de un diferencial por eje, otro diferencial al que se le denomina "central" que se encarga de repartir la potencia del motor entre los ejes delantero y trasero además de solventar el inconveniente de que cuando se toma una curva, las ruedas directrices realizan más giro que las otras, como mostraba la Figura 12. Visto esto se comprende la variedad de soluciones adoptadas a la hora de elegir los diferenciales en los vehículos, en los que además del tipo de tracción (delantera, trasera, integral o integral permanente) también influye el uso del vehículo, ya sea offroad, por carretera, competición, etc.

Seguidamente se dará una breve explicación del funcionamiento de algunos de los tipos de diferenciales de deslizamiento limitado que existen:

Diferenciales sensibles al par:

- **De discos:** En este tipo de diferenciales, un conjunto de discos similares a los de un embrague, realizados en acero endurecido y funcionando con lubricantes especiales, bloquean los dos semiejes o palieres de forma solidaria bajo determinadas condiciones de funcionamiento. La carga aplicada sobre los discos y por tanto la capacidad de hacer los semiejes solidarios se consigue mediante

dos métodos. El primero es precargando los juegos de discos mediante un muelle helicoidal o una arandela Belleville escogidos para proporcionar un valor mínimo de par que haga que se rompa la barrera estática que mantiene los ejes solidarios. La segunda, se consigue mediante un tallado de los dientes de los satélites especial, diseñado cuidadosamente para cargar los juegos de discos a medida que el par aumenta. En la Figura 17 podemos ver un diferencial de discos.

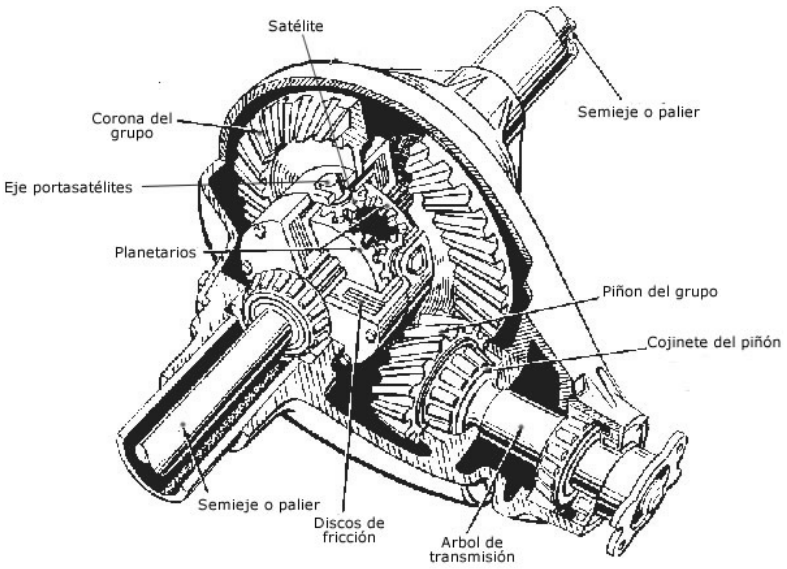


Figura 17: Diferencial de discos.

- **De engranajes o Torsen:** El diferencial torsen original usa una combinación de trenes de engranajes en donde se pueden encontrar tanto dientes rectos como con un gran ángulo de inclinación que sustituyen a los satélites de un diferencial libre convencional.

  El nombre de Torsen viene de las palabras en inglés "torque sensing" que pueden ser interpretadas como que en función del par de entrada en el diferencial pueden cambiar las condiciones de operación de éste.

  Cuando el par que entra en el diferencial es pequeño, los engranajes se encuentran poco cargados y si una rueda queda en el aire, el diferencial se comportara como un diferencial libre convencional. Es decir, en una situación donde una rueda quede en el aire o con muy poca fricción, el diferencial no logrará sacarnos. A medida que se incrementa el par, los trenes de engranajes se cargan más y de

esta forma manda más fuerza a la rueda con menos adherencia. Los principales elementos que producen la fricción necesaria para mantener solidarios los dos ejes son el propio tren de engranajes. En la Figura 18 podemos ver un diferencial de este tipo.

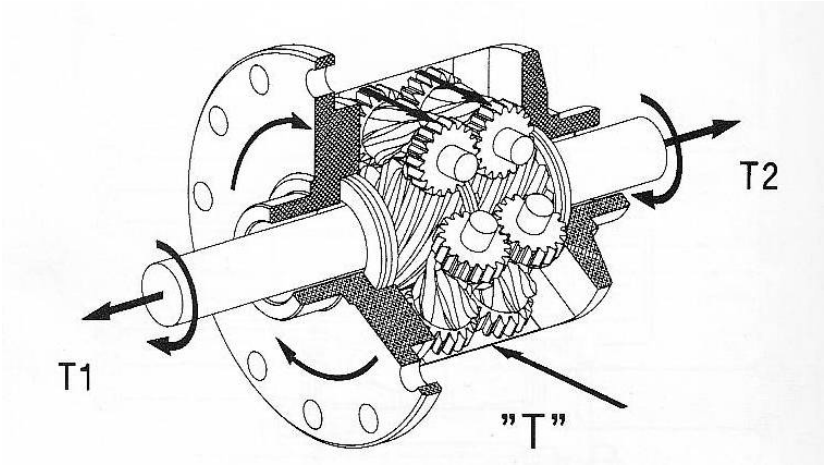


Figura 18: Esquema del diferencial tipo Torsen.

Diferenciales sensibles a la velocidad:

- **Viscosos o Ferguson:** El funcionamiento se basa en la utilización de un fluido de alta viscosidad y características especiales, normalmente silicona. Los dos semiejes o palieres están conectados entre ellos mediante un diferencial libre al que se le añaden unos juegos de discos que si bien se encuentran muy cerca unos de otros, no existe contacto entre ellos. El par transmitido es función de la diferencia de velocidades entre los semiejes y como en todos los diferenciales de deslizamiento limitado se tiende a aumentar el par que se manda a la rueda que más lenta gira.

  Mientras que la mayoría de los fluidos y lubricantes pierden viscosidad con la temperatura, el fluido utilizado en este diferencial aumenta su viscosidad cuando la temperatura sube. De este modo, los discos, mediante el movimiento relativo entre ellos, calientan el fluido y los dos semiejes se van haciendo solidarios progresivamente, esto sucede una vez que el fluido sobrepasa ligeramente los 50º C. Las pequeñas diferencias de velocidad producidas en las curvas no son

suficientes como para bloquear el diferencial de modo que trabaja casi como un diferencial libre. Solamente el deslizamiento de un neumático genera suficiente calor como para bloquear el diferencial. El aumento de temperatura y por tanto el bloqueo del diferencial tardan cierto tiempo en ocurrir, como consecuencia existe cierto retraso de respuesta ante el deslizamiento de un neumático.

Dadas sus características, este tipo de diferenciales, tiene más aplicación como diferencial central en vehículos de todo terreno 4x4 en donde también reciben el nombre de viscoacopladores. En estos casos una parte del conjunto es solidaria a las ruedas de un eje y la otra a las ruedas de otro eje. Está constituido por una carcasa solidaria al árbol de transmisión (que es uno de los ejes a los que nos hemos referido), la cual, encierra los discos. Unos discos están unidos a la carcasa y otros al portadiscos solidario al eje de salida (ver Figura 19).

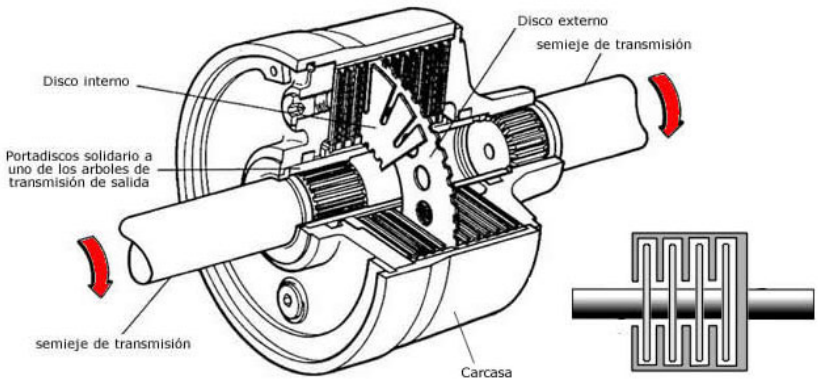


Figura 19: Esquema de un viscoacoplador.

Diferenciales bloqueables:

Estos diferenciales actúan como diferenciales libres, pero se bloquean 100 % bajo unas determinadas condiciones. Los hay que se bloquean automáticamente y también los hay que se bloquean manualmente (diferencial seleccionable). Son usados mayormente en vehículos todo terreno 4x4 en circunstancias offroad. En estos diferenciales mientras se detecta fuerza motriz en el árbol de transmisión e igual velocidad en ambas ruedas el diferencial permanece bloqueado, pero a su vez se

"desconecta" automáticamente cuando detecta que alguna de las ruedas gira a diferente velocidad (como en un viraje). Es peligroso usar este tipo de diferenciales al circular por carretera (sobre todo en circunstancias de baja adherencia como lluvia o nieve) porque en situaciones de aceleración en una curva provocarán deslizamientos al encontrarse el diferencial bloqueado y pueden causar un descontrol del vehículo y por tanto un accidente. Además aumentan el desgaste de los neumáticos y reducen la potencia del motor.

4.2.3. Utilizando los diferenciales

Todos estos tipos de diferenciales que se han visto presentan ventajas y desventajas. Así, un diferencial que tenga más bien una configuración "libre" no va a sacarnos de una circunstancia en la que una rueda se nos quede al aire o enterrada en arena (no dejándonos avanzar) y tampoco va a mejorar en exceso la dinámica de un coche de carreras al negociar una curva rápida. Por el contrario, un diferencial que presente bloqueo a baja carga, refiriéndonos con esto a situaciones de poca o ninguna aceleración, va a presentar un comportamiento dinámico "malo" a baja velocidad dada la oposición que mostrará al tomar curvas, además restará potencia al vehículo y desgastará en exceso los neumáticos.

Por lo visto anteriormente, la búsqueda de un compromiso en la configuración del diferencial es algo complejo puesto que si queremos que el mismo coche pueda pasar de estar rodando en un circuito rápido de asfalto a tener que aparcar para comprar el pan en un aparcamiento con suelo de hormigón pulido o atravesar un camino lleno de barro o nieve, es imposible que dicho vehículo "cumpla" siempre.

Ejemplos de uso de diferenciales en vehículos reales:

Vehículo de uso particular: En un vehículo de uso particular que circula principalmente por carretera y con el que no practicamos una conducción deportiva, es

decir, la elección óptima es usar un diferencial libre, ya que al presentar menos oposición al tomar curvas, produciremos menor desgaste de neumáticos y menor consumo de combustible, además de abaratar su precio porque un diferencial libre siempre va a ser más barato que uno LSD.

Vehículo de competición: Se suelen usar diferenciales de tipo LSD de discos porque mejoran la motricidad al tomar curvas al "límite". En competición, el coche más veloz no es el coche más rápido y la mayor parte del tiempo por vuelta se saca en las zonas viradas.

Vehículo de campo: Los vehículos de campo se caracterizan por circular a bajas velocidades sobre terrenos que pueden presentar poca adherencia, como barro o arena. En estos casos, debido a la superficie las ruedas pueden patinar haciendo que el vehículo no sea efectivo, por ejemplo en una pendiente hacia arriba. Por eso los más habitual es que monten diferenciales LSD bloqueables o viscosos (en el diferencial central, en el caso de tener tracción a las cuatro ruedas).

A todos los tipos de diferenciales descritos hasta ahora se les llaman diferenciales mecánicos porque el propio sistema en su conjunto consta solamente de partes mecánicas. En contraposición a estos sistemas existen los diferenciales electrónicos o activos que comentaremos más adelante.

El funcionamiento del diferencial es fundamental para entender correctamente el uso de los sistemas de control de tracción, así como de los sistemas que se describirán más adelante, como el control de estabilidad y los diferenciales electrónicos o activos.

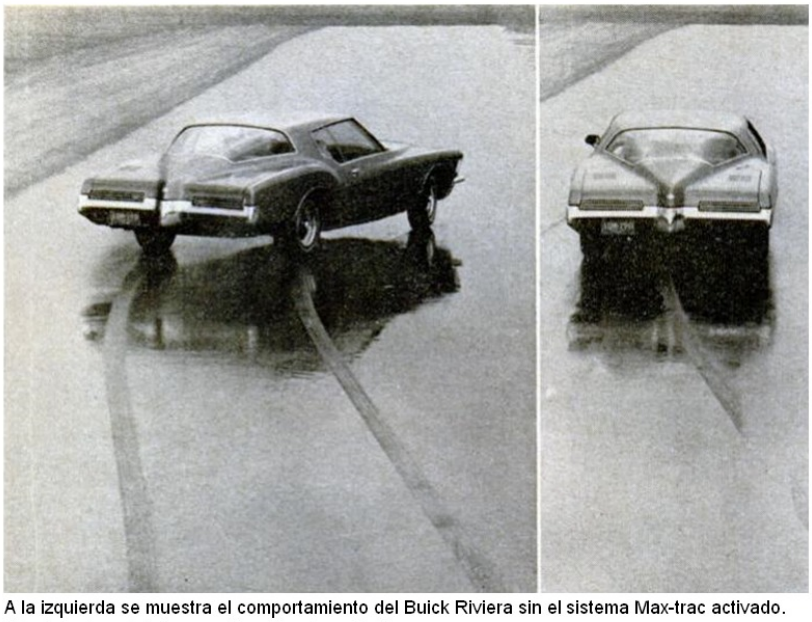
4.3. Control de tracción

El concepto de controlar la tracción no es ni más ni menos que conseguir un movimiento dinámico preciso y óptimo del vehículo sean cuales sean las circunstancias

de la superficie, velocidad o aceleración, siempre y cuando nos encontremos dentro de los límites físicos en los cuales un coche no pierda el control.

Se puede decir que los orígenes de los controles de tracción son los diferenciales de deslizamiento limitado, los cuales intentan hacer que el coche mejore sus cualidades dinámicas, como hemos visto anteriormente.

Con el desarrollo de la electrónica digital en los años 70, se crearon los primeros sistemas de control de tracción electrónicos. Por ejemplo, en 1971 la empresa Buick lanzó su primer control de tracción electrónico. Se llamaba Max-trac y consistía en leer la velocidad de las ruedas traseras y de las delanteras y si encontraba una diferencia notable entre ellas (exactamente un 10 %) reducía la potencia del motor hasta que esta diferencia bajase de un umbral definido en un 8 %. En la Figura 20 se pueden ver pruebas realizadas por Buick mostrando el funcionamiento del sistema Max-trac.



A la izquierda se muestra el comportamiento del Buick Riviera sin el sistema Max-trac activado.
A la derecha, el sistema está en funcionamiento.

Figura 20: Pruebas del sistema Max-trac en un Buick Riviera.

Los sistemas de control de tracción electrónicos más sencillos, se componen básicamente de sensores acoplados a las ruedas, los cuales estan unidos a un pequeña unidad electrónica que es el “cerebro” del sistema y que interpreta y cálcula el

deslizamiento del vehículo. Como se verá con más detalle en el desarrollo del sistema de control de tracción para el monoplaza de la Fórmula SAE, el corazón de esta unidad electrónica es un microcontrolador, un chip que realiza los cálculos y procesos necesarios. En el apartado 6.2 se explicará todo lo que necesitamos saber sobre estos chips.

Si nos preguntamos: ¿Cómo hace el sistema para disminuir la potencia del motor? La respuesta es que la unidad electrónica de control del sistema de control de tracción envía una señal a la centralita del coche, que es la que va a hacer posible disminuir la potencia del motor. Todos los coches actuales y desde hace bastante tiempo incorporan una centralita o ECU por sus siglas en inglés (Engine Control Unit) que no es ni más ni menos que otra unidad electrónica que controla el motor del vehículo. En la sección 5.2.2 se profndizará en el asunto de las centralitas electrónicas de los coches.

Con la llegada del sistema antibloqueo de frenos (ABS por sus siglas en inglés) creado por Bosch y comercializado por primera vez en 1978 en el Mercedes Clase S de la época, se abrió una vía importantísima para el desarrollo de nuevos controles de tracción más precisos y con mejores resultados en la dinámica de los vehículos, ya que los nuevos controles de tracción usarían parte de los subsistemas del ABS.

Para poder continuar con la explicación, en las siguientes líneas se explicará brevemente en qué consiste un sistema ABS y de qué partes se compone.

ABS Este sistema, lo que hace es evitar el bloqueo de las ruedas al frenar. En una frenada brusca, sobre todo en superficies resbaladizas, las ruedas de un coche equipado con frenos normales se bloquean haciendo que se pierda el control en la maniobrabilidad del vehículo además de que en casos de poca adherencia se aumenta considerablemente la distancia de frenado. No hay más que imaginarse un coche a 40 km/h sobre una gran capa de hielo. En el momento que se pisa el freno con fuerza, se bloquean las ruedas y es imposible tomar control de él. El

coche recorrerá muchos metros hasta que se pare, muchos más que si se hubiera pisado el freno de forma suave y progresiva.

El ABS evita la situación anterior en la que las ruedas se mantienen bloqueadas al estar pisado el pedal de freno. Cuando el sistema ABS detecta que hay una rueda bloqueada en una frenada, disminuye momentáneamente la presión en el circuito de frenos de la rueda bloqueada para seguidamente volver a aumentarla; esto se hace varias veces en un segundo y de esta manera se asegura que el vehículo siga siendo manejable y llegue a detenerse con rapidez y seguridad.

Entre las partes de las que se compone se encuentran unos sensores que se instalan en cada rueda los cuales emiten unas señales eléctricas proporcionales a la velocidad de giro de ésta. También necesita un dispositivo que se encargue de controlar la presión aplicada a cada una de las ruedas, este es el hidrogrupo. El hidrogrupo es controlado a su vez por una unidad de control electrónica que es la que hace los cálculos y determina cuándo debe funcionar el sistema. El esquema del sistema se puede ver en la Figura 21.

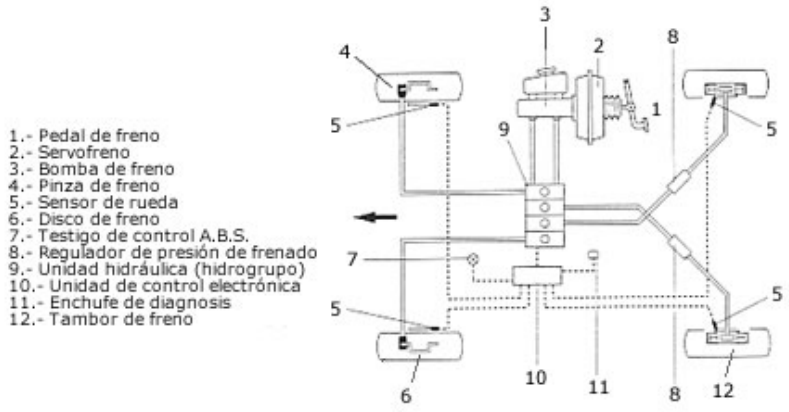


Figura 21: Esquema del circuito de frenado con ABS.

Continuando con el Control de tracción:

Se ha visto que los sensores que nos proveen la información de la velocidad a la que giran las ruedas, son indispensables en los dos sistemas (el ABS y el control de

tracción). Los nuevos controles de tracción usan el hidrogrupo del ABS para frenar en la medida necesaria cada rueda, de forma independiente.

Como ocurre con los diferenciales, hay varias formas de implementar un control de tracción y no todas ellas generan la misma respuesta, pero todas ellas se basan en las mismas acciones y éstas son las siguientes:

- Retardar o suprimir la chispa a uno o más cilindros.
- Reducir la inyección de combustible a uno o más cilindros.
- Frenar la rueda que ha perdido adherencia.

El funcionamiento de un control de tracción básico, el cual no haría uso del sistema ABS, se ha descrito anteriormente con la explicación del sistema Max-trac de Buick. Generalizandolo, tenemos que el sistema leería los sensores de las ruedas determinando la velocidad de giro de cada una de ellas, posteriormente las compararía en busca de alguna diferencia sustancial. Si el sistema detecta que una rueda gira a mayor velocidad que las demás, actúa para disminuir la potencia desarrollada por el motor hasta así subsanar esa diferencia de velocidades. Para disminuir la potencia desarrollada por el motor se efectúan dos de las acciones que ya se habían visto:

- Retardar o suprimir la chispa a uno o más cilindros.
- Reducir la inyección de combustible a uno o más cilindros.

Gracias al desarrollo del sistema ABS, podemos llevar a cabo el frenado individual de las ruedas del vehículo, esto es sorprendentemente útil para mejorar la capacidad de tracción del coche puesto que nos permite simular la respuesta de un diferencial de deslizamiento limitado, lo que conlleva mejoras en situaciones donde un control de tracción básico no puede ayudarnos, como por ejemplo las siguientes:

- En una salida desde parado, si una de las ruedas está sobre una placa de hielo, al acelerar, esta rueda comenzará a patinar por acción del diferencial (hablando de un diferencia libre) y la fuerza motriz disminuirá al verse "escapada" por esta rueda que patina. Por mucho que aceleremos no conseguiremos que la rueda traccione y tampoco conseguiremos aumentar el par desarrollado ya que esta rueda girará prácticamente sin oposición de resistencia. En este caso en concreto, si conseguimos frenar esta rueda conseguiremos que el motor pueda desarrollar más par mandando más fuerza a la rueda que sí tiene adherencia y por tanto comenzando el avance del coche.

- Cuando aceleramos en una curva en un coche con tracción en un solo eje, por ejemplo un tracción trasera, la rueda exterior carga más peso, por tanto es la rueda con mayor adherencia. Sin embargo la rueda interior, descargada de peso, tiene menor adherencia provocando que pueda llegar a patinar en un momento "límite". Para aumentar ese "límite" podemos frenar ligeramente la rueda interior, haciendo que aumente el par en la rueda exterior y consiguiendo con todo ello un mayor "agarre" y por consiguiente una mejor motricidad al tomar la curva.

Resulta palpable cómo de efectivo puede ser añadirle al control de tracción la capacidad de frenar individualmente una rueda, pero las cosas no son tan fáciles y a la hora de programar la unidad de control electrónica se deben tener en cuenta algunas consideraciones:

1. Tocar el freno cuando se produce un deslizamiento circulando a altas velocidades puede suponer un gran riesgo, pudiendo bloquearse una rueda y ocasionar la perdida del control del vehículo y por consiguiente provocar un accidente. Por otro lado frenar cuando se va a alta velocidad provoca un calentamiento excesivo en los frenos y también un desgaste de éstos. En estos casos, cuando por ejemplo en una curva el coche comienza a presentar deslizamiento, es conveniente únicamente

reducir la potencia desarrollada por el motor, hasta que a través de esta acción consigamos volver a una situación neutra.

2. Otro factor a tener en cuenta es que en una salida desde parado, si por ejemplo una rueda motriz está en el aire o ha quedado enterrada, cosa que ocurre con frecuencia en conducción offroad, no nos va a servir de nada que al detectar deslizamiento en esta rueda actuemos sobre el motor disminuyendo la potencia, ya que no conseguiremos salir de esta situación. En este caso sí que habría que frenar esa rueda que gira "loca" para así mandar más fuerza a otra rueda.

Vistas estas consideraciones, un control de tracción debe ser "programado" o configurado de forma que funcione correctamente en varias situaciones, por ello, generalmente se da mayor importancia a la acción de frenado cuando se circula a baja o muy baja velocidad. Cuando se circula a partir de una cierta velocidad, unos 40 km/h, se suprime la posibilidad de accionar el freno, actuando solamente para disminuir la potencia del motor, como se ha comentado anteriormente.

4.4. Más sensores

Para determinar cuándo debe actuar un sistemas electrónicos de control de tracción se usan sensores de velocidad en las ruedas, pero los datos proporcionados por éstos pueden no ser suficientes o mejorarse sustancialmente añadiendo otros tipos de sensores. Los sistemas más modernos incorporan más tipos de sensores como por ejemplo sensores de aceleración transversales (también llamados sensores G) que miden la fuerza lateral, muy útiles para determinar si al tomar una curva existe mayor o menor riesgo de pérdida de adherencia. También se incluyen sensores que determinan el ángulo de giro del volante, revoluciones del motor y posición de giro del cigüeñal.

4.5. Diferencial electrónico o activo

Con las funcionalidades que nos proveen los controles de tracción electrónicos y con la descripción de su funcionamiento advertimos que a veces se comportan como si de un diferencial de deslizamiento limitado se tratasen, por se pueden desarrollar sistemas electrónicos que emulen el funcionamiento de éstos y a los cuales llamamos diferenciales activos o diferenciales electrónicos. Si en un mismo coche instalamos un control de tracción, un diferencial activo y ABS, contribuiremos a aumentar su seguridad y a mejorar su funcionamiento.

A través de diferentes configuraciones de la unidad electrónica de control podremos desarrollar diferenciales activos que se comporten de manera diferente según nuestras necesidades. Comprenderemos esto mejor cuando se analice, en el siguiente apartado, algunos de los diferentes sistemas electrónicos comerciales que son los que incorporan los vehículos producidos en serie.

4.6. Sistemas comerciales

Con tantos tipos de configuración del control de tracción y habiendo tantas marcas fabricantes de automóviles que los montan, tenemos un amplio abanico de acrónimos para definir los diferentes controles de tracción: ASR, TCS, EDS, XDS, etc. Además, por si fuera poco, distintos fabricantes llaman de igual manera a sus sistemas, pero el funcionamiento no tiene porqué ser exactamente igual, así por ejemplo varias marcas llaman a su control de tracción ASR y no todos actúan de la misma forma.

Vamos a analizar algunos de los sistemas actuales:

ASR: Son las siglas de Automatic Stability Control o Anti-Slip Regulation. En algunas marcas este control de tracción actúa solamente disminuyendo potencia al

motor cuando el momento lo requiere, sin embargo en otras marcas también actúa frenando la rueda que ha perdido adherencia a baja velocidad o partiendo desde parado. Entre otras marcas, Volkswagen y Mercedes usan este acrónimo.

DTC: Son las siglas de Dynamic Traction Control. Es nombre que usa BMW para su control de tracción. Este sistema de BMW limita la alimentación del motor y frena las ruedas de forma selectiva para evitar la pérdida de tracción. Depende del modelo de BMW existen pequeños cambios en la configuración pero básicamente el sistema permite un cierto patinamiento de las ruedas motrices para que la conducción de estilo deportivo pueda ser más ágil, pudiendo llegar a un ligero sobreviraje siempre que no se circule por encima de 70 km/h o que los sensores registren aceleraciones transversales superiores a 0,4 g (en ese momento, el sistema entiende que el sobreviraje es excesivo y puede crearse una situación de peligro).

EDL: Lo monta Volkswagen en alguno de sus modelos. Volkswagen lo define como Bloqueo Electrónico del Diferencial y permite arranques suaves y confortables sobre firmes que presentan niveles de adherencia desiguales. Si una rueda empieza a patinar, el bloqueo electrónico del diferencial frenará la rueda en la medida necesaria, dirigiendo potencia a la rueda con la mayor tracción. El bloqueo del diferencial electrónico reduce el desgaste de los neumáticos y está activo hasta una velocidad de aprox. 40 Km/h (80 Km/h en los vehículos equipados con 4MOTION, la tracción a las cuatro ruedas de Volkswagen).

XDS: Este es el nuevo sistema electrónico que incorporan los coches deportivos de Volkswagen. Lo monta por primera vez el VW Golf GTI de la sexta generación (2009). Es un diferencial electrónico de deslizamiento limitado que proporciona presión de frenado a la rueda interior en una curva (entre 5 y 15 bar) para evitar que patine. El resultado en el VW Golf GTI es una magnífica motricidad, con muy leves pérdidas de tracción.

4.7. Sistema de control de estabilidad

El sistema de control de estabilidad también llamado ESP (del alemán "Elektronisches Stabilitätsprogramm" – Programa electrónico de estabilidad) por el nombre comercial que se le dio cuando salió al mercado, supuso un paso más en el desarrollo de la seguridad en situaciones de perdida de control en curvas. Mejora sustancialmente lo que ya hacía el control de tracción en estas situaciones, que era básicamente reducir la potencia del motor hasta que la situación fuera controlable. El ESP es una funcionalidad que se añade al control de tracción, sistema que a su vez utiliza el sistema ABS.

El ESP comenzó a desarrollarse en una colaboración entre Bosch y Mercedes-Benz sobre el año 1987 cuando ya había una base considerable en cuanto a los demás controles electrónicos como el ABS y el ASR, ya que se nutre de ellos para su funcionamiento. No se comercializó hasta 1995, momento en el que se montó por primera vez en un coche de producción en serie, el Mercedes Clase S de la época.

El ESP es un fuerte sistema de seguridad ya que literalmente consigue que el coche siga la trazada correcta aún cuando ocurre un deslizamiento de las ruedas. Esto lo consigue gracias a los numerosos sensores que necesita para funcionar, como sensores de velocidad en las ruedas, sensores que determinan el ángulo de giro del volante y sensores de aceleración lateral.

Cuando tomamos una curva demasiado rápido con un vehículo de tracción delantera, éste tiende al subviraje, que significa que el coche tiende a ir recto describiendo un giro bastante menor del que marcan las ruedas directrices. Esto provoca un deslizamiento que detecta rápidamente el ESP. En cuestión de milisegundos el sistema evalúa la situación realizando numerosos cálculos hasta determinar la trayectoria correcta que debería seguir el vehículo y en ese momento a parte de disminuir la potencia del motor, cosa que haría el control de tracción, frena las ruedas individualmente de forma totalmente controlada para que el coche siga la trayectoria

calculada. ¿Cómo es esto posible? Por ejemplo, si el coche tiende a irse de frente cuando debería describir un giro a la derecha, el sistema frena el tren delantero con más intensidad en la rueda interior para hacer que éste "se meta" en la trayectoria. Algo análogo sucede en los vehículos con tracción trasera, salvo que estos coches en lugar de tender a irse rectos, tienden a "irse de atrás" haciéndoles girar más de lo que marcan las ruedas directrices; a esto se le llama sobreviraje. En la Figura 22 se puede ver la comparación de un vehículo con ESP y otro sin ESP al esquivar un obstáculo.

Figura 22: Comparación entre un vehículo con ESP y otro sin él.

Como nota interesante, no está de más comentar el dato conseguido por la propia Mercedes en su centro de simulación de Berlín. Mercedes realizó una prueba, en la que participaron 80 conductores de ambos sexos, en recorridos en los que aparecían curvas con placas de hielo. Los 80 conductores superaron la prueba sin accidentarse con el coche equipado con ESP. Sin embargo, sin el ESP el 78 % de ellos sufrieron un accidente.

4.8. Unión de sistemas

En los coches modernos se aúnan uno o varios de los sistemas descritos hasta ahora, consiguiéndose así mejorar enormemente la seguridad y funcionamiento del vehículo en

cualquier circunstancia. Por poner un ejemplo, el nuevo Volkswagen Touareg (Figura 23) equipa un diferencial central de deslizamiento limitado mecánico tipo Torsen y los siguientes sistemas electrónicos: ABS + ESP + ASR + EDL.

Figura 23: Volkswagen Touareg.

5. Objetivos y descripción básica del sistema

5.1. Introducción al sistema empleado

Una vez conocidos los diferentes sistemas de control de tracción y lo que se puede hacer con ellos, se describirá el sistema a diseñar y posteriormente implementar en el monoplaza de Fórmula SAE realizado por el equipo UPMRacing.

Como reza el título de este libro, se desarrollará un sistema de control de la tracción en curva y salida desde parado, con el objetivo de mejorar los siguientes aspectos:

- Mejora de la aceleración en la salida desde parado.
- No perder el control del vehículo al acelerar en la salida de las curvas.
- Aumentar la seguridad del piloto.
- En general, ir más rápido en circuito.

Por tanto, la misión es disminuir la potencia del motor cuando se presente una situación de pérdida de tracción o deslizamiento. Es necesario para ello diferenciar entre dos tipos de situaciones:

1. **Salida desde parado o muy baja velocidad:** Cuando aplicamos una fuerte aceleración, ya sea desde parado o a muy baja velocidad corremos el riesgo de que debido a la gran potencia del coche las ruedas traseras, que son las motrices, comiencen a patinar. Si la adherencia de los dos neumáticos es similar, las dos ruedas pueden comenzar a patinar, en cambio, si una de las ruedas está sobre una superficie mucho más resbaladiza que la otra, lo que ocurrirá es que sólo ésta patinará. En ambos casos el avance del coche se verá afectado en mayor o menos

medida, dependiendo sobre todo del nivel de adherencia de los neumáticos con el suelo y de la aceleración transmitida.

2. **Curva a media-alta velocidad:** En este caso, juega, además de las variables que teníamos en caso anterior: adherencia y aceleración, la variable velocidad, que indudablemente al tomar un giro repercutirá en generar una fuerza lateral en el vehículo que puede provocar que éste pierda tracción. Esto es así, porque los neumáticos disponen en un momento dado de un nivel de adherencia y si la fuerza que ejercemos sobre la rueda supera este nivel de adherencia, la rueda patinará o deslizará hasta que estas fuerzas se vuelvan a igualar. Este sistema no es un ESP y lógicamente no salvará situaciones en las que se efectúe un giro a una velocidad mucho mayor de la adecuada, entre otras cosas porque el sistema sólo actúa restando potencia al motor y no corrigiendo la trayectoria del vehículo como lo haría un ESP. Lo que sí que conseguirá el sistema será mitigar el sobreviraje ocasionado por una excesiva entrega de potencia del motor al salir de una curva, sobre todo si el pavimento está mojado o con poca adherencia.

Básicamente se trabajará sobre estas dos situaciones, diseñando el sistema para que el vehículo no pierda la tracción cuando se llegue al límite. Al llegar al límite en el cual el coche empieza a presentar el deslizamiento, reduciremos la potencia generada por el motor, de este modo la velocidad disminuirá llegando de nuevo al estado en el que los neumáticos puedan, con su nivel de adherencia, dirigir el vehículo sin deslizamiento.

Para implementar el control de tracción, se hará uso de unos sensores acoplados a un sistema electrónico, que detectará el momento en el que el coche pierde la tracción y actuará en el motor para reducir la potencia. Como se vio en los diferentes sistemas descritos en el apartado 4.3 y con el desarrollo de la electrónica digital desde los años 70, el uso de ésta se ha generalizado en la realización de prácticamente todas las unidades de control en los sistemas del vehículo. La unidad electrónica que se diseñará, será una placa de circuito impreso que constará de varios chips y componentes electrónicos,

entre ellos, reguladores de tensión eléctrica y un microcontrolador, que es el "cerebro" de todo el sistema. Se hablará de microcontroladores en la sección 6.2.

Entrando un poco en el aspecto de la reducción de la potencia del motor, aunque es un tema en el que entraremos más adelante en el apartado 6.3, la unidad electrónica de control, mandará una señal a la centralita del vehículo, que será la encargada de alterar ciertos parámetros en el funcionamiento del motor para reducir la potencia entregada por éste. En el apartado 5.2.2, se hablarán más cosas sobre las centralitas en los coches.

En cuanto a los sensores, se usarán 4 sensores de efecto Hall que están acoplados uno a cada rueda y que nos servirán para leer de ellos una señal proporcional a la velocidad de giro de cada una de ellas. Haremos llegar estas señales a la unidad electrónica de control por medio de unos cables. La unidad hará los cálculos necesarios para determinar si existe deslizamiento y generará la señal que se aplicará a la centralita del coche.

El sistema también contará con un panel de mandos que será la interfaz con el piloto, donde podrá apagar/encender el control de tracción entre otras cosas. Finalmente, el sistema se debe de poder conectar a un bus de comunicaciones llamado Bus CAN, con el objetivo de poder ser configurado para la modificación de ciertos parámetros como diámetro de las ruedas, deslizamiento permitido por el sistema, etc.

De momento hemos conformado una idea bastante básica, pero aproximada, del sistema y podemos ver un esquema en la Figura 24. En la sección 6 se irá desarrollando más.

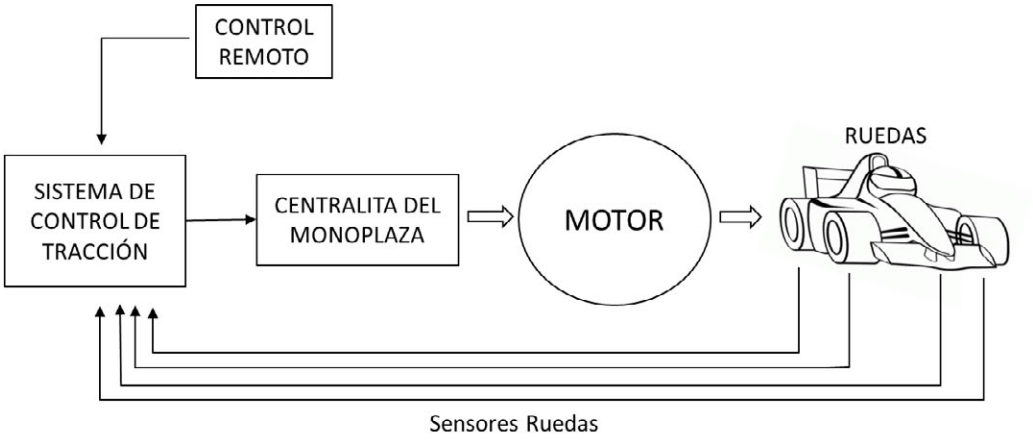


Figura 24: Diagrama del sistema.

5.2. Especificaciones del coche

Para hacernos una idea de las prestaciones del coche de Fórmula SAE fabricado por el equipo UPMRacing, vamos a conocer algunas de sus especificaciones. Además en la Figura 25 podemos ver una foto del UPM007, el coche fabricado en el año 2009-2010.

Generalidades	
Aceleración de 0 a 100 km/h	4 *segundos*
Velocidad máxima	180-200 km/h
Peso total	230 kg.

Motor	
Tipo	Yamaha R6
Número de cilindros	4
Cilindrada	600 cc
Potencia	80 cv
Par	60 Nm
Admisión	Fibra de carbono con restricción de 20 mm para la entrada de aire

Electrónica	
Centralita	Performance Electronics
Sensores	- Temperatura del agua
	- Apertura de la mariposa
	- Sensor del cigüeñal
	- Sensor de efecto Hall en cada rueda

Sistema de frenos	
Pedalier	3 pedales
Pinzas	Pinzas de dos pistones AP Racing
Discos de freno	Perforados de 260 mm de diámetro:
	- 2 frenos de disco delanteros (uno por rueda)
	- un único trasero unido al diferencial
Llantas	13”

Transmisión	
Embrague	Mecánico
Cambio	Secuencial de 6 velocidades
Diferencial	Torsen
Relación primaria cigüeñal caja de cambios	1,995
Relación piñon - corona	13:42

Suspensión	
Neumáticos	20.5x6-13 R25B Hoosier/ 20x7.5-13 R25B Hossier (delantero/trasero)
Tipo	Doble trapecio con barra estabilizadora
Batalla	1550 mm
Vía delantera / vía trasera	1168 / 1143 mm

Figura 25: Monoplaza UPM007

5.2.1. Sensores de efecto Hall

Como hemos podido ver en las tablas de especificaciones, el coche posee un sensor de efecto Hall acoplado a cada rueda. Estos sensores son punto clave en la realización de nuestro sistema porque con ellos vamos a determinar la velocidad de cada una de las ruedas.

El sensor de efecto Hall o simplemente sensor Hall o sonda Hall (denominado según Edwin Herbert Hall) se sirve del efecto Hall para la medición de campos magnéticos o corrientes o para la determinación de la posición.

Efecto Hall Cuando por un material conductor o semiconductor, circula una corriente eléctrica, y estando este mismo material en el seno de un campo magnético, se comprueba que aparece una fuerza magnética en los portadores de carga que los reagrupa dentro del material, esto es, los portadores de carga se desvían y agrupan a un lado del material conductor o semiconductor, apareciendo así un campo eléctrico perpendicular al campo magnético y al propio campo eléctrico generado por la batería. Este campo eléctrico es el denominado campo Hall, y ligado a él aparece la tensión Hall.

Haciendo uso del efecto Hall, un sensor de este tipo tiene varias aplicaciones:

- Mediciones de campos magnéticos (Densidad de flujo magnético)
- Mediciones de corriente sin potencial (Sensor de corriente)
- Emisor de señales sin contacto
- Aparatos de medida del espesor de materiales

En nuestro caso, disponemos de un disco dentado de material metálico que va acoplado de forma solidaria al eje de cada rueda y de un sensor de efecto hall que va fijo y que también se encuentra en cada rueda. Cuando la rueda gira, también lo hace el disco dentado y cuando esto ocurre cada uno de los dientes pasa cerca del sensor generando así una determinada diferencia de potencial (ver Figura 26). Este fenómeno nos va a permitir contar el número de pulsos de la señal que genera el sensor, en un determinado espacio de tiempo.

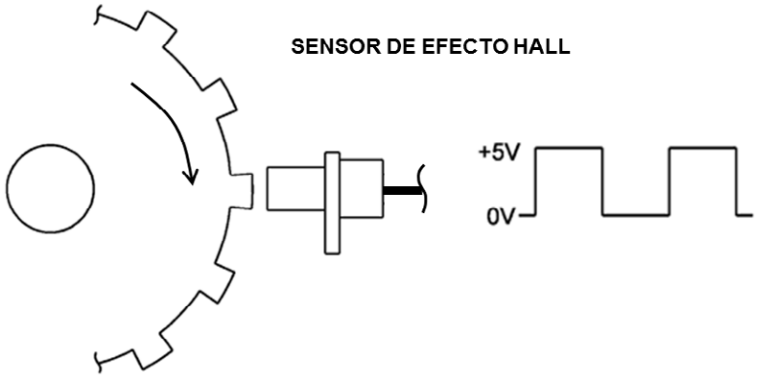


Figura 26: Sensor de efecto Hall.

Concretamente, el sensor de efecto Hall utilizado es un sensor de la marca Honeywell modelo 1GT101DC.

5.2.2. Centralita: qué es y para qué se usa

La centralita electrónica también se conoce como unidad control de motor o ECU (las siglas de Engine Control Unit). Es un dispositivo electrónico que controla, calcula y actúa en una serie de parámetros en un motor de combustión para lograr un funcionamiento correcto y óptimo (mayor potencia y menor gasto de combustible), esos parámetros son, entre otros: control de la inyección de combustible, punto óptimo de la chispa de ignición (curva de avance de encendido), presión del turbo, etc. En los coches más modernos también se incluyen dentro de la centralita, la gestión de los sistemas ABS, Airbags, Control de tracción y otros controles electrónicos que se incorporan en los coches actuales, haciéndolas más y más complejas cada vez.

En lo referente al control del motor, las ECU determinan y calculan las señales y parámetros del motor, monitoriándolo a través de diversos sensores, éstos incluyen: sensor MAP (presión del colector de admisión), sensor de posición del acelerador, sensor de temperatura del aire, sensor de oxígeno, sensor de velocidad de rotación del cigueñal y otros.

La ECUs empezarón su basta incursión en los motores en los años 70 con la proliferación de los avances en la electrónica y sus mejoras son sustanciales ya que proveen al motor de un sistema de control de sus parámetros, más fiable, preciso, efectivo y configurable. Hasta entonces eran los sistemas mecánicos los encargados de gestionar estos parámetros de funcionamiento del motor.

No es el propósito de este libro profundizar en conceptos de mecánica del automóvil, pero sí se explicará el concepto de “avance de encendido” en el motor de combustión ya que es algo que cualquier persona interesada mínimamente en la automoción (y que no lo supiera ya) agradecerá, como yo lo he hecho al estudiarlo para escribir estas líneas. Además nos dará una visión clara de la mejora y flexibilidad que aportan las centralitas electrónicas a los motores.

Sistema de avance de encendido

De una forma muy somera y reducida, el funcionamiento de un motor de combustión se basa en llenar un cilindro con una mezcla de aire y combustible, comprimir esta mezcla empujando un pistón que se encuentra dentro del cilindro y después generar una chispa para que se inflame. Esta combustión, empuja el pistón hacia abajo generando un movimiento. Usando uno, o combinando varios cilindros y acoplando a cada pistón una biela y al extremo de éstas un cigüeñal, conseguimos un movimiento de rotación que es el que se usa, en el caso de un coche, para producir el movimiento de giro de las ruedas. En las figuras 27 y 28 se pueden observar mejor las partes básicas del motor de combustión y el nombre que se le dan.

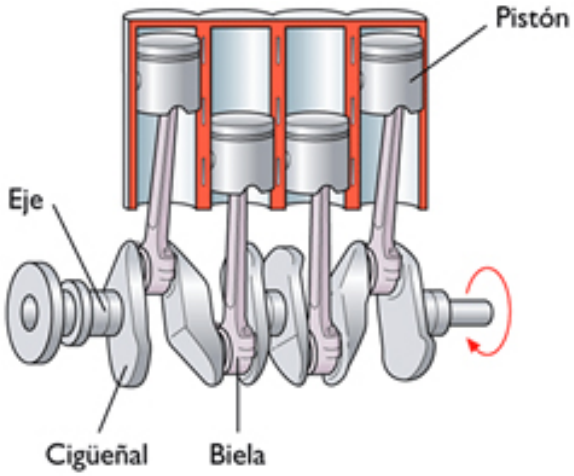


Figura 27: Vista transversal de las partes de un motor.

También es necesario mencionar que es lo que se conoce como PMS (punto muerto superior) para entender lo que supone el "sistema de avance de encendido". El PMS, es el momento en el que el pistón se encuentra lo más dentro posible del cilindro, por así decirlo es su final de carrera superior. Por el contrario, el PMI (punto muerto inferior) es justo lo contrario, el final de carrera inferior del pistón, momento en el que se encuentra lo más fuera posible del cilindro. Ver figura 29.

Siguiendo con esto, teóricamente existe un momento óptimo en el cual queremos que se inflame la mezcla y es justo cuando el cilindro ha llegado al punto más

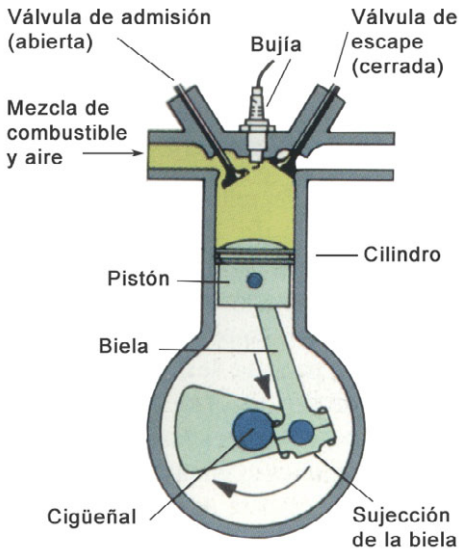


Figura 28: Partes del motor detalladas.

alto, el PMS, así se genera la inflamación en el punto de mayor compresión de la mezcla y en el momento que más recorrido hacia abajo tiene el pistón. De este modo, el motor generará más potencia y gastará menos combustible. El problema surge en que la inflamación de la mezcla de combustible y aire no es instantánea y transcurre un tiempo desde que producimos la chispa en el cilindro hasta que el frente de llama va recorriendo la mezcla e inflamándola. Realmente tampoco queremos que la explosión sea exactamente cuando se esté en el PMS, sino justo después, ya que en el PMS el pistón y la biela están en línea y en ese punto no hay un ángulo suficiente como para que la fuerza generada en la inflamación de la mezcla pueda hacer rotar el cigüeñal.

Es por ello que se necesita un "avance" en la generación de la chispa de encendido para sincronizar el momento óptimo de la combustión de la mezcla y logar un funcionamiento más eficiente en términos de potencia y gasto de combustible. Este "avance de encendido" se mide en grados y representa el giro del eje del cigüeñal, el cual se adelanta al momento en el que llega el PMS. Ver figura 30.

Considerando que el tiempo de inflamación de la mezcla es siempre el mismo, debemos tener en cuenta que el tiempo que el cigüeñal tarda en girar unos grados varía a diferentes rpm del motor (revoluciones por minuto), por lo que si queremos

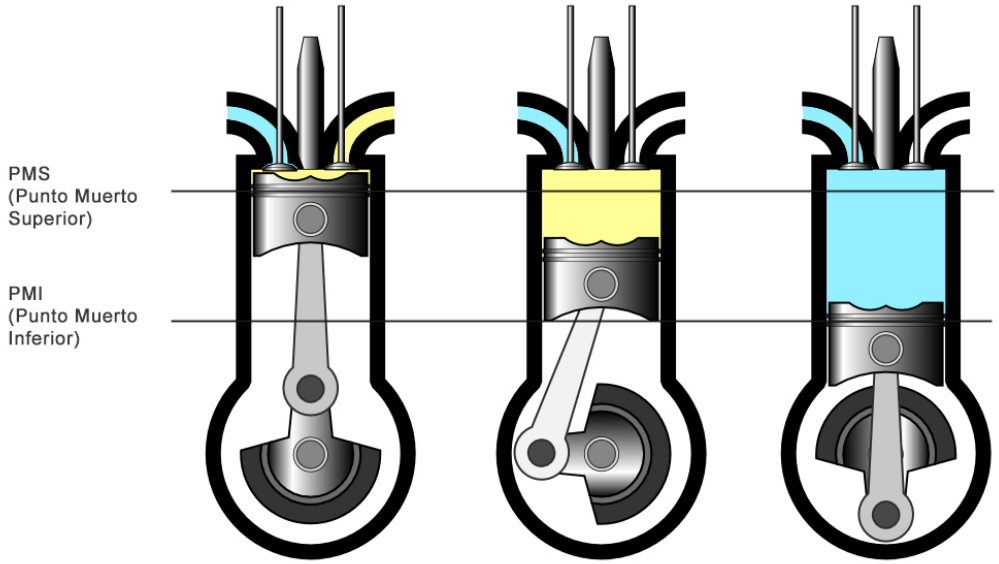


Figura 29: Posiciones PMS y PMI.

sincronizar la chispa de encendido con el momento óptimo de llevarla a cabo, el "avance de encendido" debe variar junto con las rpm. A mayores rpm, mayor deberá ser el avance de encendido. Además de todo esto, el tiempo de inflamación de la mezcla varía en función de la presión del colector de admisión, del porcentaje de mezcla aire/combustible, de la temperatura, etc. lo que conlleva la necesidad de crear un sistema de avance de encendido, el cual es complicado de poner a punto y que genere unos valores diferentes de avance dependiente de todos los factores anteriormente expuestos: lo que se conoce como la "curva de avance de encendido".

Antes de la llegada de la electrónica en los sistemas de control de los parámetros del motor, se tenían diversos sistemas mecánicos. En concreto, para controlar el avance de encendido se añadían al distribuidor dos sistemas: el regulador centrífugo (que aporta el avance de encendido referente a la velocidad de giro del motor) y el regulador de vacío (aporta el avance de encendido dependiente de la presión de vacío del colector de admisión). Estos elementos eran propensos al desgaste, desajustes, fallo mecánico y se limitaban a proporcionar una única

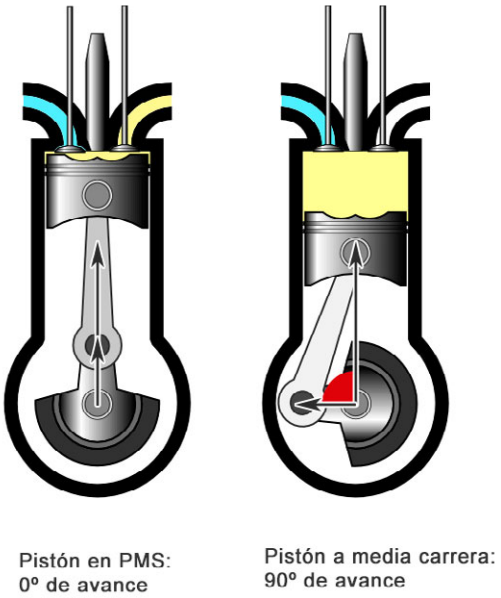


Figura 30: Representación del avance en un motor.

"curva de avance de encendido" que no podía ser modificada sin modificar físicamente los componentes de dichos elementos.

Todo esto que he expuesto referente al avance de encendido también es aplicable a otro parámetro importante del motor: la inyección de combustible. No se profundizará en este aspecto, pero en la inyección de combustible es importante tener una correcta mezcla de combustible/aire, además de introducir más o menos cantidad de mezcla en el cilindro. El porcentaje de mezcla aire/combustible es dependiente de factores como la posición de la mariposa del acelerador, la temperatura del motor, su velocidad de giro, la presión absoluta en el múltiple de admisión, la cantidad de oxígeno en los gases de escape... en resumidas cuentas, este es otro aspecto que conlleva un sistema complejo para su correcto funcionamiento.

Volviendo al tema de las ECU, nuestro coche de Fórmula SAE incluye una centralita de la marca Performance Electronics modelo PE-ECU-1 que se puede ver en la figura 31.

Figura 31: Centralita PE-ECU-1.

La PE-ECU-1 es un potente sistema de programación de inyección y encendido en tiempo real que está diseñado para controlar la mayoría de motores de explosión de 4 cilindros. De una forma general el sistema optimiza el rendimiento del motor mediante:

- Control de curva de encendido.
- Control del sistema de inyección.
- Múltiples opciones configurables (hasta 8 entradas/salidas analógicas y digitales).

La PE-ECU-1 proporciona la adquisición de datos del motor y permite acceso en tiempo real para maximizar las prestaciones y localizar fallos en el vehículo en funcionamiento. Daremos sus características de una forma más completa en las siguientes tablas.

Características Generales
Control de encendido e inyección para motores de 1,2,4,6 y 8 cilindros
Motores 2 y 4 tiempos
Protección de picos de tensión.
Software de programación bajo Windows, fácil e intuitivo
Construcción robusta, incluso estanca para aplicaciones náuticas
Tamaño compacto: (120mm Largo X 133mm Ancho X 54mm Alto)
Generación automáticas de mapas base para arranque del motor
Comunicación RS232 soportando traductores USB-SERIE.
Sistema de adquisición de datos a memoria de PC
Interpolación lineal en todas las tablas
Representación gráfica en tiempo real

Control de Alimentación y Encendido
Compensaciones de arranque, temperatura de aire, temperatura de refrigerante, enriquecimiento por aceleración, tensión de batería, corte en retención.
Control sobre el ángulo de apertura de inyector
Control de carga por posición de mariposa o presión en colector
Encendido por “chispa perdida” o distribuidor para motores de 4 cilindros
Ajuste de retardo de bobina
Compensación por temperatura de aire
Control de máximo régimen motor
No requiere módulos de encendido externos. Dispara las bobinas desde la ECU

Entradas
Sensores estándar GM
Presión en colector
Posición de la mariposa
Temperatura de aire
Temperatura de batería
Posición de cigüeñal
3 entradas analógicas para la modificación del encendido o inyección
2 entradas digitales para corte de alimentación, encendido o apagado de motor.

Salidas
4 drivers de saturación para inyectores de alta impedancia y 2 drivers para bobinas
Control sobre la bomba de combustible
4 digitales configurables basados en Rpm, posición de mariposa, presión de colector, temperaturas de aire y refrigerante. Salida de tacómetro.

Las aplicaciones más frecuentes para el uso de esta centralita son las siguientes:

- Control de inyección y encendido en motores modificados.
- Conversión de motores de carburación a inyección.
- Aplicaciones de carrera de cualquier tipo
- Tareas de investigación y desarrollo y uso educacional en universidades y escuelas técnicas
- Como equipo de origen en automóviles y motocicletas.

Además, el control y modificación de los parámetros de la centralita se hace mediante un software instalado en un PC el cual da información y facilita las labores de configuración. En las figuras 32 y 33 se pueden ver algunas capturas de pantalla.

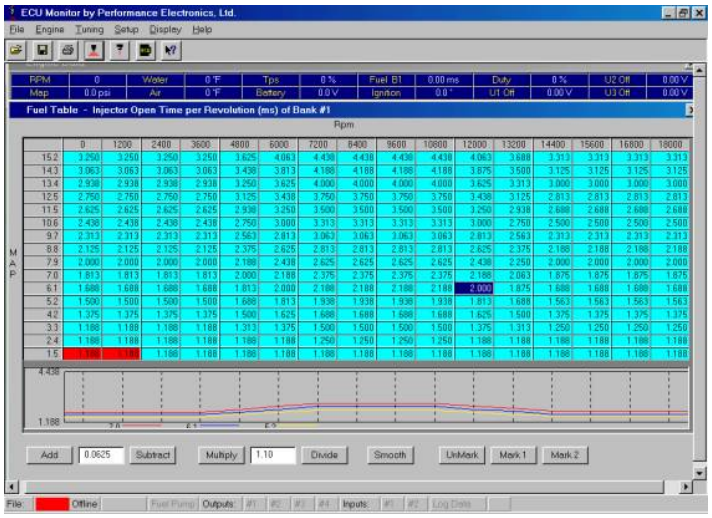


Figura 32: Mapa de inyección

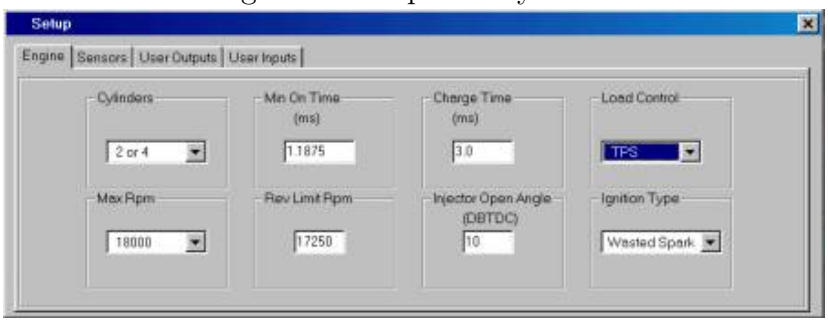


Figura 33: Pantalla de configuración de motor

El equipo UPMRacing se encarga de la correcta configuración y gestión de la centralita del monoplaza.

En este proyecto necesitamos hacer uso de la centralita para la ya comentada disminución de potencia que debemos generar cuando en el vehículo exista una pérdida de tracción. Para este cometido se usará una de las entradas de señal disponibles en la PE-ECU-1.

La unidad electrónica de control de tracción que se está diseñando mandará una señal a la ECU cuando detecte deslizamiento para que ésta disminuya la potencia en

el motor. Los diferentes métodos de disminución de potencia, así como la forma en la que vamos a conseguir esta disminución, se tratarán en el apartado 6.3

6. Diseño del sistema

6.1. Control de un proceso: El regulador

6.1.1. Sistema de control

No es el objetivo de este apartado profundizar en el conocimiento de las teorías de control, por lo que no nos vamos a extender mucho en dar a conocer todas las formas de control de un sistema ni las bases teóricas, simplemente nos limitaremos a comentar algunas cosas básicas en el uso de sistemas de control y posteriormente la forma de control elegida para el sistema que vamos a diseñar.

Bien, de forma conceptual un sistema de control está definido como un conjunto de componentes que pueden regular su propia conducta o la de otro sistema con el fin de lograr un funcionamiento predeterminado, de modo que se reduzcan las probabilidades de fallos y se obtengan los resultados buscados.

Fijándonos ahora en el ámbito de la ingeniería, la teoría de control, también llamada ingeniería de control o regulación automática, estudia el comportamiento de los sistemas dinámicos[1], tratándolos como cajas o bloques con una entrada y una salida. Entre otras cosas se focaliza en modelar matemáticamente una gama diversa de estos sistemas dinámicos y el diseño de controladores que harán que estos sistemas se comporten de la manera deseada. Aunque tales controladores no necesariamente son electrónicos y por lo tanto la ingeniería de control es a menudo un subcampo de otras ingenierías como la mecánica.

[1]Un sistema dinámico es un sistema cuyo estado evoluciona con el tiempo. El comportamiento en dicho estado se puede caracterizar determinando los límites del sistema, los elementos y sus relaciones; de esta forma se puede elaborar modelos que buscan representar la estructura del mismo sistema.

En una primera clasificación, se pueden diferenciar los sistemas de control según su comportamiento:

1. **Sistema de control de lazo abierto:** Es aquel sistema en que solo actúa en el proceso con la señal de entrada y da como resultado una señal de salida independiente a la señal de entrada, pero basada en la primera. Ejemplos: Llenar un vaso de agua con un grifo automático de un lavabo. Al pulsar el botón que tiene el grifo sale agua durante un determinado tiempo, este sistema no controla que el vaso se haya llenado o esté rebosándose. Otro ejemplo podría ser un sistema de alumbrado que se encienda a una hora prevista de la tarde y se apague a una hora prevista por la mañana.

   Estos sistemas se caracterizan por:

   - Ser sencillos y de fácil concepto.
   - Nada asegura su estabilidad ante una perturbación.
   - La salida no se compara con la entrada.
   - Ser afectado por las perturbaciones.
   - La precisión depende de la previa calibración del sistema.

2. **Sistema de control de lazo cerrado:** Para evitar los problemas del control en lazo abierto, la teoría de control introduce la realimentación. Un regulador de lazo cerrado utiliza la realimentación para controlar los estados y las salidas de un sistema dinámico. El nombre de "lazo cerrado"hace referencia al camino que sigue la información en el sistema: la salida del proceso se mide y cuantifica mediante un sensor y se combina de cierta forma con la entrada para que vuelva a actuar sobre el proceso.

   El control con lazo cerrado presenta las siguientes ventajas sobre el control en lazo abierto:

   - Corrección de las perturbaciones (tales como rozamiento impredecible en un motor).

- Buen comportamiento incluso con incertidumbre en el modelo, es decir, en aquellos casos en que la estructura del modelo no representa perfectamente la realidad del proceso o los parámetros del modelo no se pueden medir con absoluta precisión.
- Permite estabilizar procesos inestables.
- Tolerancia a variaciones en los parámetros.

Para convertir en lazo cerrado los sistemas que anteriormente se pusieron de ejemplo, deberíamos, en el caso del grifo, poner un sensor en el vaso (un aforador) que detecte cuándo el agua superta un cierto nivel y en consecuencia actuar para cerrar el grifo. En el caso del sistema de alumbrado, se podría poner un sensor de luminosidad para que se detecte cuando sea de noche y cuando sea de día y actuar en consecuencia.

Para nuestro propósito, vamos a emplear una estrategia de control basada en un regulador PID. El regulador PID probablemente sea el diseño de control más empleado, por ser el más sencillo respecto a las ventajas que nos reporta. "PID"son las siglas de Proporcional-Integral-Derivativo, y se refiere a los tres términos que operan sobre la señal de error para producir una señal de control. Esto lo veremos seguidamente.

6.1.2. Regulador PID

Se intentará explicar de una forma sencilla y no muy extendida el funcionamiento de un regulador PID en el control de un proceso.

Podemos intuir que el objetivo de cualquier estrategia de control es mantener una variable (llamada controlada) próxima a un valor deseado (conocido como punto de ajuste). La forma de regulación de un PID se basa en el concepto de lazo cerrado que ya hemos visto. En la figura 34 veríamos su diagrama de bloques.

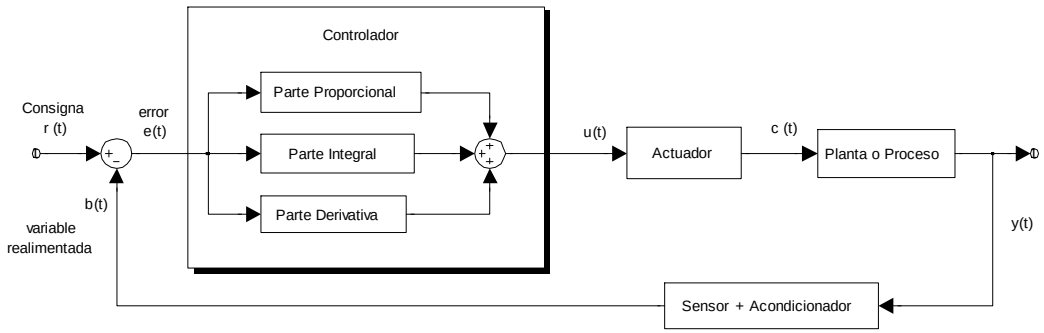


Figura 34: Diagrama de bloques de un Regulador PID

El valor que queremos alcanzar en nuestro proceso es representado por una señal que se conoce como **consigna** o **señal de referencia**. En el ejemplo de un sistema de calefacción, la consigna será una variable o señal que represente la temperatura que queremos alcanzar (punto de ajuste). Esta señal puede ser por ejemplo un voltaje o el valor de un registro, ya que estos sistemas trabajan en la mayoría de los casos en el mundo analógico o digital.

Al ser un sistema en lazo cerrado es necesario disponer de un **sensor** para leer y cuantificar la salida del proceso a controlar y que sirva como realimentación. En el ejemplo del sistema de calefacción, este sensor leería la temperatura actual de una estancia. Más bien, el sensor leería una señal que usaríamos para interpretar la temperatura. Es necesario en este punto acondicionar la señal del sensor para que sea compatible con la señal de consigna, es decir, que sea de la misma magnitud y rango que la señal consigna para poder compararlas en igualdad de condiciones.

Se conoce como **error** del sistema a la diferencia entre la señal medida y ya acondicionada del sensor y la señal de referencia. Así se determina en cada instante la diferencia que hay entre el valor deseado y el valor medido. Lo deseado, por tanto, es que la señal de error valga cero.

Tenemos también dentro del conjunto de bloques del sistema, un **controlador** que en base a la señal de error, genera una salida que va a ser la encargada de llevar al

proceso al valor deseado. Para convertir la señal generada por el controlador en una magnitud física que haga variar el proceso (la variable manipulada) se necesita lo que llamamos un actuador. El **actuador** genera la señal de entrada en el proceso con el fin de modificar de forma controlada la variable que queremos regular. En el ejemplo esta variable sería la temperatura en una habitación, el actuador sería el circuito que hace pasar corriente por una resistencia para calentarla y el controlador el bloque genera la señal que gobierna al actuador. Normalmente disponemos de un actuador que genera una señal de diferente naturaleza, rango o magnitud que la señal de salida del controlador. En estos casos es necesario acondicionar la señal de salida del controlador.

Por ultimo tenemos un bloque que representa el proceso a controlar al que le llamamos planta. Como apunte importante, decir, que uno de los objetivos en la ingeniería de control es conocer la forma matemática que rige el funcionamiento de la planta, conocida como función de transferencia, el acto de regular el sistema es más fácil de esta manera porque permite conocer a priori la salida del proceso frente a una entrada dada. Por supuesto, al tratar prácticamente siempre con plantas o procesos analógicos del mundo real, existen multitud de variables que "modifican" eventualmente la función de transferencia y parametrizar estas variables, que se conocen como perturbaciones, para que formen parte de la función de transferencia, es muy difícil, a veces imposible, dada su naturaleza. En el ejemplo de la calefacción pueden surgir varias perturbaciones como la apertura de una puerta o ventana en la estancia, el frío o calor en el exterior, la humedad, el número de personas en la estancia, etc.

Esta estrategia de control de sistemas mediante el regulador PID llega a un compromiso entre facilidad de implementación y efectividad al controlar nuestro proceso.

Como resumen, vamos a relacionar todas las partes y variables del regulador con el sistema de calefacción que hemos estado tratando como ejemplo:

- La **planta** o proceso será la temperatura de una habitación dada.

- La **señal de consigna** será una señal analógica que representa la temperatura a la que queremos llevar la estancia. Por ejemplo se puede cambiar mediante el potenciómetro de un mando. Entrada: el potenciómetro accionado por una persona. Salida: el valor de la señal que genera el potenciómetro y que es la referencia de temperatura.

- El **sensor** indispensable para la realimentación del sistema se coloca en la habitación y lee una señal proporcional a la temperatura actual de la estancia. Entrada: temperatura de la habitación. Salida: valor de la señal proporcional a la temperatura leída.

- El **actuador** será un circuito que manda corriente eléctrica a una resistencia para calentarla.

- El **controlador** será un circuito que en base a la diferencia entre la señal del sensor y la señal de consigna genera una señal que gobierna el actuador.

Lo realmente importante y difícil de calcular en estos sistemas es el controlador, más concretamente la forma en la que el controlador genera la señal que atacará al actuador. Esta señal describirá el comportamiento del proceso y lo llevará a los valores deseados. **Un buen controlador tiene que ser capaz de llevar a la planta a los valores deseados de forma estable y en un tiempo reducido**.

Centrándonos ahora en el controlador de un regulador PID, recordamos lo dicho unos párrafos antes «"PID" son las siglas de Proporcional-Integral-Derivativo, y se refiere a los tres términos que operan sobre la señal de error para producir una señal de control».

El controlador PID genera la señal de control en base a la expresion (1).

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt} \tag{1}$$

Vemos aquí claramente los tres sumandos de los que se compone esta señal, los cuales representan cada uno una acción de control diferente que sumadas generan la señal de control. Cada término que opera en esta fórmula tiene un peso en la señal de salida que cobra mayor importancia dependiendo de la naturaleza del error (además de las constantes que multiplican cada termino, por supuesto).

Para saber cómo se ha llegado a la anterior expresión basta con conocer la forma de actuar de esta estrategia de control, la cual atiende y se centra en los tres aspectos más importantes de la naturaleza del error:

1. La cantidad de error
2. El promedio del error o error acumulado durante el tiempo
3. Velocidad de cambio del error

Cada uno de estos tres aspectos se puede cuantificar y calcular matemáticamente.

1. Para responder ante la cantidad de error en un momento dado basta con multiplicar el error por un número, obteniendo una señal proporcional al error. De ahí viene su nombre: acción proporcional. Esta acción sólo actúa cuando existe error actual en el sistema.
2. La operación matemática que denota el promedio de una señal durante el tiempo, es la integral. De aquí también viene el nombre de acción integral.
3. Y por último la derivada es la operación que puede cuantificar la velocidad de cambio de una señal. Esta parte se conoce, por tanto, como acción derivativa.

Como a mí en su día me costó un poco entender la parte de acción integral y derivativa, simplemente porque no supe identificar rápidamente los conceptos de

integral y derivada en este contexto, me gustaría explicarlo un poco y perdonad si para algunos es algo obvio y trivial.

Integral

La integral, o más bien, la integral definida, no es otra cosa que el área de la figura geométrica que se genera entre una señal y el eje x de coordenadas. Si dibujamos en un eje de coordenadas un señal de voltaje en corriente continua de 3 voltios que se prolonga en el tiempo tendremos lo que se ve en la Figura 35. La integral de esta señal desde su comienzo hasta el segundo 3 será el área que vemos rayada en la misma figura.

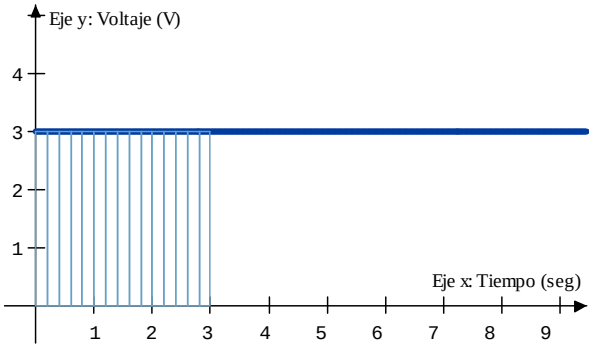


Figura 35: Señal de voltaje constante de 3 voltios.

Este área es muy fácil de calcular porque la figura geométrica que describe es un cuadrado, así que multiplicando base por altura tenemos que la integral de esta señal en el periodo $0 < t < 3$ vale $3 \cdot 3 = 9$. Lo siguiente es ver cómo en este caso, conforme va pasando el tiempo, si vamos calculando en cada instante el valor de la integral de esta señal desde su comienzo ($t = 0$), ésta no hace más que aumentar. Cuando el área está por debajo del eje de coordenadas x, se dice que es negativa. Así pues, si la señal fuera -3 voltios en corriente continua, el área sería negativa en todo momento.

Con este fácil ejemplo sacamos varias conclusiones:

1. El valor de la integral en un periodo de tiempo es positivo si la señal es positiva.

2. El valor de la integral en un periodo de tiempo es negativo si la señal es negativa.

3. El valor de la integral crece en valor absoluto con el tiempo si la señal es constante.

4. El valor de la integral de una señal nula es cero.

5. Si una señal en un periodo dado ha tomado valores positivos y negativos, el área generada bajo su curva, corresponde a áreas positivas y negativas y el valor de la integral será la resta de estas dos áreas. Ver Figura 36 donde está sombreada la integral de una señal eléctrica sinusoidal desde $t = 0$ a $t = 5$.

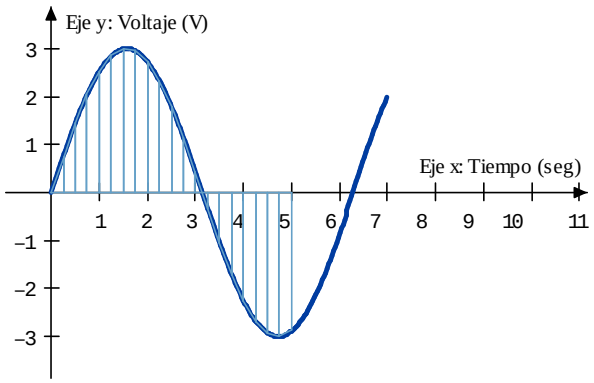


Figura 36: Señal sinusoidal con valores positivos y negativos.

Centrándonos en la acción integral del regulador, cuando en un sistema la señal de error se mantiene en el tiempo, la acción integral crece excitando así al proceso para reducir dicho error. Y más importante, **la acción integral es capaz de mantener un valor de excitación en el proceso aun cuando el error ya se ha hecho nulo**, cosa que en determinados procesos es fundamental por ser procesos que necesitan de un aporte constante de energía para mantener constante el punto de ajuste. Por ejemplo en el control de velocidad de un automóvil, cuando ajustamos la velocidad a 100 km/h el sistema debe mantener una determinada inyección de combustible constantemente.

Derivada

La derivada de una función en un punto, representa la pendiente de la recta tangente a la función que pasa por ese punto. En el caso más fácil, una señal constante, la pendiente vale cero en todos sus puntos. Aumentando un poco la dificultad, en una señal tipo rampa, o sea una recta con pendiente, ésta también se mantiene constante en todos sus puntos, pero el valor sería diferente de cero. Para calcular la pendiente de esta señal, es decir, su derivada, basta con coger un periodo corto de tiempo y dividir el incremento en su valor entre el tiempo empleado. En la Figura 37 lo podemos ver. He usado una señal de voltaje en corriente continua como ejemplo.

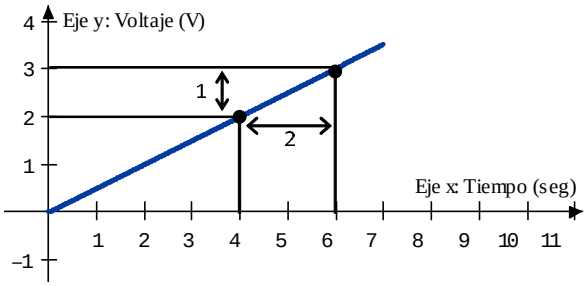


Figura 37: Representación de la derivada de una señal tipo rampa creciente.

En este caso la derivada valdría $(1 \div 2) = 0{,}5$ y como hemos anotado antes, permanece constante en el tiempo. Si la rampa decreciera con el tiempo, la pendiente y por consiguiente la derivada, sería negativa.

En una señal que aumenta con el tiempo de forma no constante, la derivada en el instante actual se iría incrementando. Esto se puede ver bien gráficamente al ir trazando restas tangentes al punto en el cual queremos ver la derivada y viendo que su pendiente crece. Podemos ver esto en la Figura 38 donde trazamos la recta tangente a una señal creciente tipo parabólica en los instantes $t = 2$ y $t = 6$.

Centrándonos en la acción derivativa, vemos que calculando la derivada de la señal tenemos una magnitud proporcional al crecimiento o decrecimiento de la señal, que es

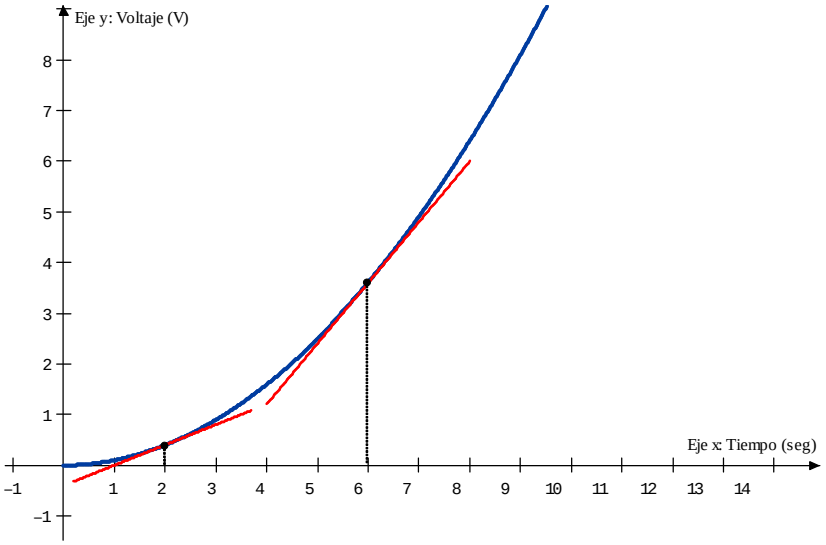


Figura 38: Representación de la derivada en dos puntos de una señal tipo parabólica.

justo lo que representa la acción derivativa en la estrategia de control PID: actuar en base a la velocidad de cambio del error.

Expuesto lo anterior, quiero hacer mención a un texto que he leído en un trabajo (bibliografía [24]) que me parece muy interesante en esto que estamos viendo sobre el control de un sistema:

> *Cada proceso tiene una dinámica propia, única, que lo diferencia de todos los demás; es como la personalidad, la huella digital de cada persona, como su ADN... Por lo tanto, cuando en un Lazo de Control sintonizamos los algoritmos P (Proporcional), I (Integral) y D (Derivativo) de un Controlador, debemos investigar, probar, compenetrarnos con la 'personalidad' del proceso que deseamos controlar, debemos medir, calibrar y mantener todo tipo de variables de proceso, y sintonizar los parámetros de los algoritmos de control.*

Existe una extensa bibliografía sobre técnicas de "sintonización" de parámetros en reguladores PID que sientan las bases teóricas sobre los cálculos de estos parámetros para hacer la regulación proporcional-integral-derivativa lo más eficaz posible, pero

aquí no las veremos. Lo que sí vamos a hacer es analizar meticulosamente el proceso que nos ocupa sobre el control de tracción para "sintonizar" de forma adecuada el sistema obteniendo así unos resultados satisfactorios.

6.1.3. Saturación integral o Windup

En un sistema controlado con un regulador que posea acción integral puede ocurrir un fenómeno que se denomina satuación integral o windup y que hace que el valor de la acción integral sea tan grande que sature la acción de control y esté presente aun cuando el error del sistema ha desaparecido.

Pongamos un ejemplo para explicar el fenómeno de windup. Imaginemos que tenemos un horno en el cual controlamos su temperatura con un regulador PI. El calor que suministra la resistencia que calienta el horno para que aumente la temperatura, está limitada a un máximo, por lo que la acción de control presenta unos límites físicos a la hora de actuar en el sistema, en este caso, el máximo voltaje de funcionamiento de la resistencia del horno. Si nos encontramos una perturbación en el sistema, por ejemplo abriendo la puerta del horno en un día muy frío, la temperatura disminuiría del punto de consigna, encontrándonos un error en el sistema. En este caso, la acción proporcional actuaría inmediatamente para corregirlo, pero el sistema no puede llegar a la temperatura deseada, haciendo que la acción integral aumente indefinidamente mientras se encuentre la perturbación. Si cerramos la puerta del horno, la temperatura comenzaría a subir, disminuyendo el error y disminuyendo por tanto la acción proporcional, pero la señal de control seguiría saturada a causa de la acción integral, que solo disminuirá su valor al encontrar error negativo, esto es, cuando la temperatura del horno haya superado la temperatura de consigna, haciendo que el sistema presente una sobreoscilación y que no se estabilice hasta que la acción integral no vuelva a valores lineales (por debajo de la saturación).

Queda claro que no podemos dejar que la acción integral aumente indefinidamente, existiendo algunas técnicas anti-saturación que corrigen este fenómeno. Algunas de ellas son:

Banda integral

En este método lo que se hace es anular la acción integral si el valor de la variable manipulada se encuentra por encima o por debajo de un margen en torno al set point. Se hace para que una vez la perturbación ha terminado, la acción integral no se continúe aplicando cuando se supere este margen, evitando que la variable manipulada aumente mucho su valor, alejándose del set point, hasta que el error negativo haga disminuir la acción integral, generándose en este proceso una gran sobreoscilación, que suele durar un tiempo proporcional al tiempo de duración de la perturbación.

La desventaja de este método es que no elimina el tiempo de oscilación hasta que la variable manipulada vuelve a controlarse, pero sí limita los valores máximo y mínimo de la variable manipulada en este tiempo de oscilación.

Tampoco se puede hacer la banda integral muy pequeña puesto que reduciremos la velocidad con la que el sistema vuelve al set point, al dejar la acción de control solo controlada por la acción proporcional. Dicha acción proporcional, cuando nos encontramos cerca del set point (teniendo errores bajos), tiende a ser muy pequeña. Esto se agrava si nuestro proceso requiere de un aporte constante de energía para mantener el set point.

Parada de la suma

En este método el valor de la suma de la acción integral es congelado cuando el regulador está en saturación y su valor permanece constante mientras el regulador está en saturación.

Con este método, acotamos el valor de la acción integral cuando el regulador ya está haciendo todo lo posible por disminuir la señal de error. Pero este método no se libra de tener un exceso de valor de acción integral cuando la perturbación ha desaparecido, lo cual retrasa la recuperación del control cuando desaparece la perturbación.

Una modificación a este método es hacer cero el valor de la acción integral cuando el valor de la acción proporcional no esté actuando en la zona lineal. Cuando la acción proporcional vuelve a actuar en la zona lineal, la acción integral vuelve a comenzar su suma del error. Esto contraresta un poco el aumento del valor de la acción integral cuando existen cambios bruscos del punto de consigna o en presencia de pertubaciones.

Resta integral

La idea de este método es que el valor de la acción integral sea decrementado cuando el controlador entra en saturación por una cantidad proporcional al exceso de la señal de control, por encima de un valor máximo. Cuando el controlador deja de estar en saturación, la acción integral aumenta en presencia de error. Con este método se mantiene en unos valores moderados el valor de la acción integral en presencia de perturbación y saturación del controlador. Es necesario ajustar la constante que multiplica el exceso de señal de control para que la señal de control generada no se vuelva inestable y caiga por debajo del máximo en presencia de perturbación.

En nuestro sistema usaremos un algoritmo para no saturar la señal de control a causa del valor de la acción integral, pero la planta que queremos controlar (motor-ruedas-deslizamiento) tiene un comportamiento específico que tenemos que estudiar y veremos mejor en el apartado 6.8.

6.2. Qué son los Microcontroladores y por qué usar uno

Un microcontrolador no es ni más ni menos que un pequeño chip que dentro tiene las cosas necesarias para funcionar como un computador: un procesador, memoria, y unidades de entrada/salida. El microcontrolador es el cerebro de un sistema electrónico y se encarga de hacer cálculos y generar señales en base a unas directrices que previamente se han de grabar en él. El microcontrolador suele ir soldado en una pequeña placa de circuito impreso (PCB, de sus siglas en inglés Printed Circuit Board) y acompañado de otros componentes electrónicos para que pueda funcionar, tales como reguladores de tensión para la alimentación, buffers para aislar las señales eléctricas, etc. Con un microcontrolador somos capaces, por ejemplo, de mantener la temperatura en un frigorífico, hacer funcionar un cronómetro digital o escuchar música mp3.

Se han comentado en la sección 5.2.2 que habla de la centralita del vehículo, las mejoras tan grandes que introducen los sistemas electrónicos en el control de procesos, ya que nos proveen de una herramienta fiable, precisa, configurable, de bajo coste y con bajo consumo energético. En definitiva, es prácticamente una obligación usar un microcontrolador para poder desarrollar nuestro sistema.

Se usará un microcontrolador en el sistema para medir la velocidad de las ruedas y analizarlas para así detectar deslizamiento, para comunicarnos con el piloto a través de una interfaz (los mandos), para generar una señal analógica que enviaremos a la centralita del vehículo y que así se reduzca la potencia del motor y para mandar y recibir información a través de un bus de comunicaciones llamado Bus CAN.

¿Qué modelo elegir? Hay varias empresas que fabrican microcontroladores y multitud de modelos: los hay más potentes que realizan mayor número de cálculos por segundo, los hay con diferentes “carcasas” y tamaños, llamados encapsulados, y los hay también con mayor número de periféricos y módulos embebidos dentro del microcontrolador. Para este proyecto se nos recomendó por parte del departamento de sistemas electrónicos y de control la utilización de un microcontrolador del

fabricante *Microchip* modelo *dsPIC33FJ128*, el cual aúna gran velocidad de ejecución, tamaño reducido con un encapsulado apto para montaje superficial en la placa de circuito impreso y gran cantidad de periféricos y módulos integrados: módulo para comunicaciones Bus CAN, muchos Timers, controlador PWM (modulador por ancho de pulso), etc. Además, en la escuela se dispone de todo el kit de desarrollo que nos ayudará a programar el microcontrolador en cuestión.

Las tareas a ejecutar por parte del microcontrolador son grabadas en el microcontrolador en un proceso previo llamado programación. En este paso de programación, es cuando se diseñan algoritmos de control que se traducen al lenguaje concreto que entiende el microcontrolador y después se graba en su memoria para que la unidad de control ejecute secuencialmente estos pasos una y otra vez.

¿Cómo usarlo? Para usar el microcontrolador debemos generar un código escrito en un lenguaje de programación que podamos grabar dentro del microcontrolador para que este lo interprete y por supuesto dotarlo de alimentación eléctrica y de las conexiones mínimas necesarias.

De la alimentación eléctrica y su correcto montaje y conexionado ya nos ocuparemos más adelante en la sección 8.

Para escribir el código o instrucciones necesarias para que funcione el microcontrolador usamos un software específico del fabricante, este software se llama *MPLAB IDE*. Además tenemos que instalar un "añadido" a ese software, esto es un compilador que transformará el código que escribimos mediante sentencias en instrucciones de más bajo nivel que son las que entiende el microcontrolador a la hora de su programación. Como escribiremos el código del programa en un lenguaje de programación de alto nivel: lenguaje C, el compilador que debemos "añadir" al software *MPLAB IDE*, es el *compilador C30*.

Con el software *MPLAB IDE* podremos escribir el programa en lenguaje C que deberá ejecutar el microcontrolador, además de poder testearlo y transferirlo a la memoria de éste. Para transferir el programa que hemos escrito en el entorno MPLAB al microcontrolador, usaremos un pequeño dispositivo de *Microchip* llamado *MPLAB ICD-2*, este dispositivo, además de transferir el programa al microcontrolador, nos permitirá leer el valor que toman las variables de nuestro programa para poder testear su funcionamiento.

Ahora vamos a detallar un poco por encima las características del microcontrolador del que disponemos y seguidamente las características específicas que se van a usar de él para poder realizar el sistema de control de tracción.

Características generales del microcontrolador
- Ejecuta hasta 40 millones de instrucciones por segundo (MIPS)
- Tiene un oscilador interno
- Tamaño de la palabra de datos de 16 bits
- Tamaño para la palabra de instrucciones de 24 bits
- Acceso de memoria directo (DMA) con 8 canales
- Hasta 63 fuentes de interrupción
- Hasta 85 pines digitales de entrada/salida
- 9 timers
- Módulo Output Compare, que sirve entre otras cosas para generar una señal PWM
- Módulo Input Capture, que sirve entre otras cosas para medir el periodo de una señal
- Módulos de comunicaciones I2C, UART, SPI, ECAN
- Convertidor analógico a digital (ADC)
- Empaquetado de 100 pines TQFP

Básicamente haremos uso de los siguiente módulos del microcontrolador para que el sistema funcione:

Módulos que usaremos
- 4 canales del módulo Input Capture que usaremos para medir el periodo de la señal generada por los sensores de efecto hall acoplados a cada rueda - 2 timers. Uno de ellos es usado en conjunción con el módulo Input Capture y el otro lo usaremos para establecer el periodo de análisis del sistema, generando una interrupción cada cierto tiempo. - El módulo Output Capture para generar una señal PWM que nos servirá para atacar a la centralita del vehículo y que ésta disminuya la potencia del motor. - El módulo de comunicaciones ECAN usado para leer y escribir datos al sistema, tales como diferentes mapas de motor y cambio de parámetros.

6.3. Métodos para la disminución de potencia

Después de lo expuesto en el apartado 4.3 donde hablábamos del control de tracción y en el apartado 5.2.2 de la Centralita, podemos decir que los métodos para poder disminuir la potencia del motor son los siguientes:

- Suprimir la chispa de encendido en los cilindros.
- Retardar la chispa de encendido en los cilindros.
- Reducir la inyección de combustible en uno o más cilindros.

Aplicar cada uno de estos métodos tiene sus ventajas e inconvenientes además de sus particularidades a la hora de implementarlos. Se explicará de forma general cada uno de ellos.

Suprimir la chispa de encendido en los cilindros

Suprimir la chispa de encendido durante un tiempo supone que la mezcla no se inflame en el cilindro y por consiguiente no habremos generado fuerza para que el pistón y la biela muevan el cigüeñal, haciendo que durante ese tiempo no se genere potencia en el motor y que el avance del vehículo quede a merced de la inercia. Es un método sencillo y fiable y que bien ajustado funciona correctamente pero tiene como inconveniente que los gases de la mezcla de aire y gasolina sin quemar salen por el tubo de escape.

Ventajas:

1. No tenemos mezcla pobre en los cilindros.
2. Es un método relativamente sencillo de implementar y que no causa daños ni alteraciones importante en el motor.
3. Haríamos uso de una entrada digital de la ECU facilitando la generación de la señal desde nuestro sistema.

Inconvenientes:

1. La mezcla de aire y gasolina sin quemar salen por el tubo de escape.

Retardar la chispa de encendido

Vimos en el apartado 5.2.2 de la centralita, qué significaba el avance de encendido y porqué era necesario. En definitiva, es necesario para sincronizar el punto óptimo en el que ocurra la combustión de la mezcla de aire y combustible, para así obtener mayor potencia y menor consumo. Si retrasamos el avance, generando así la chispa de encendido un tiempo después del óptimo, después de que el cilindro haya empezado su carrera hacia abajo, podemos obtener una disminución de potencia.

Ventajas:

1. No tenemos mezcla pobre en los cilindros.

2. No tenemos mezcla sin quemar por el tubo de escape.
3. Resulta en un control preciso de la potencia del motor.

Inconvenientes:

1. La disminución de potencia no es muy grande por lo que solo funcionará con deslizamientos pequeños.
2. Es más difícil de implementar y ajustar. Tendríamos que usar una señal analógica para atacar a la ECU, cosa que hace más complejo el diseño de nuestro sistema.
3. Es necesario controlar bien el avance de la chispa de encendido para que no haya problemas con el motor.

Reducir la inyección de combustible en uno o más cilindros

Este método tiene varias interpretaciones y además es algo dependiente de la tecnología de inyección empleada en el motor, como pueden ser la inyección indirecta y la inyección directa.

De forma general, en un motor de combustión interna de gasolina de 4 tiempos, el aire entra dentro del cilindro en el tiempo de admisión cuando el pistón está en fase de descenso. Hay que pensar que el aire entra en el cilindro a causa del vacío generado por el pistón al bajar y entra a través del colector de admisión. El aire que entra es mezclado con el combustible en unas proporciones muy concretas, que si no se cumplen causan malas combustiones de la mezcla y pueden generar problemas de funcionamiento. El inyector es la pieza que expulsa gasolina para mezclarla con el aire que entra al cilindro. La mezcla más eficiente, para el caso de la gasolina, es de entorno a 15:1 (15 partes de aire por una de gasolina). Para hacer la mezcla correctamente debemos de saber cuanto aire está entrando en el cilindro y esto depende de varios factores, pero sobre todo de la posición de la mariposa del acelerador. Cuando la mariposa está totalmente abierta el cilindro se llena de más aire (y por tanto el inyector debe inyectar más gasolina).

A priori, podemos pensar que disminuyendo la cantidad de combustible que expulsa el inyector, se llenaría con una mezcla que tiene menos combustible siendo la combustión mucho menos potente y así podríamos disminuir la potencia, pero **no podemos hacer esto** porque generaríamos lo que se llama una mezcla pobre (con baja cantidad de combustible) y esto puede provocar graves problemas en el motor, como sobrecalentamiento en la combustión (por el exceso de oxígeno en la mezcla) y combustiones expontáneas (lo que también se conoce como autoencendido) y pistoneo (inflamación de la mezcla antes de llegar al PMS con lo que se genera una fuerza contraria a la de avance, generando problemas mecánicos) ¿Qué se hace entonces cuando se dice que se reduce la inyección de combustible? Pues se hace de dos formas:

1. Una de ellas es tomando el control de la mariposa de aceleración. Si disponemos de una mariposa de aceleración electrónica que pueda ser controlada por la centralita, podemos "quitar" el control de la posición de la mariposa al acelerador (la abertura de la mariposa estaría gobernada por la posición del acelerador) y cerrarla total o parcialmente (simulando así que "levantamos el pie del acelerador") y disminuyendo el aire que entra al cilindro y por tanto el inyector expulsará menos cantidad de combustible para hacer la mezcla correcta. En nuestro caso no podemos hacer esto porque la centralita no gobierna la posición de la mariposa del acelerador de forma electromecánica.

2. La otra forma es cortando totalmente la inyección en uno o más cilindros durante todo el tiempo de admisión del cilindro. Ya hemos dicho que generar una mezcla pobre es malo para el funcionamiento del motor, pero "no generar"mezcla, es decir, suprimir la inyección de combustible durante el tiempo de admisión y que solo entre aire al cilindro sí es algo que podemos hacer y que funciona correctamente para disminuir la potencia. En dispositivos de control de tracción comerciales, lo que se suele hacer es suprimir la inyección en la admisión de uno o más cilindros durante un ciclo, dejar varios ciclos un funcionamiento normal y volver a hacerlo, así se reduce

la potencia de forma controlada. En nuestro caso, tenemos dos problemas: por un lado no podemos elegir a qué cilindros aplicar esta supresión temporal en la inyección de combustible a través de la ECU y deben ser a todos ellos y por otro lado, para suprimir totalmente la inyección en todo el tiempo de admisión de un cilindro, debemos disponer de alguna señal para conocer la posición del cilindro en todo momento o al menos cuando se suceden los PMI y PMS.

Ventajas:

a) No tenemos mezcla pobre en los cilindros.

b) No tenemos mezcla sin quemar por el tubo de escape.

c) Si podemos elegir los cilindros a los cuales aplicarles este método, puede resultar un control preciso de la disminución de potencia.

Inconvenientes:

a) Es un sistema más complicado de implementar que además requiere un sensor a parte para conocer la posición del cilindro.

Para elegir el método más correcto para el sistema, además de hacer todo este estudio, se contactó con el equipo UPMRacing para que se aconsejara cual era la forma más factible de abordar el problema. Después de hablarlo, llegamos a la conclusión de que lo mejor era un método lo más sencillo y directo posible. De esta forma **se ha elegido el método de la supresión de chispa**. Sopesamos el inconveniente de que cuando el control de tracción entre en acción, la supresión de chispa ocasionara una mezcla sin quemar por el escape, pero por lo hablado con el equipo, es algo con lo que se puede convivir en un coche de carreras y que no generará a la larga mayor problema en el motor.

Ahora es necesario ajustar bien la frecuencia y ciclo de trabajo de la señal que atacará la ECU para suprimir la chispa, para que reduzca poca potencia del motor cuando haya presente poco deslizamiento y más potencia cuando el deslizamiento

aumente, además de hacerlo de forma suave y progresiva. Esto se estudiará en el apartado 6.7 del PWM.

6.4. Cálculo de la velocidad

En este apartado se describe la forma en la que se mide la velocidad del coche, cosa que requiere un doble análisis, por un lado, la medición de la velocidad de cada una de las ruedas y por otro, la forma de disponer de una medida global y fiable de la velocidad del coche. Esto último servirá para aplicar diferentes correcciones para disminuir la potencia del motor según la velocidad a la que vaya el coche.

6.4.1. Medición de la velocidad de cada rueda

Sabemos que cada rueda del vehículo lleva acoplada un disco dentado que gira junto con ella, y también un sensor, que va fijado cerca de los dientes del disco y que genera, cada vez que un diente pasa al lado de él, un pulso. Estos pulsos son los que nos permitirán calcular la velocidad a la que va cada rueda.

El objetivo es medir la velocidad de giro de las ruedas con una resolución y precisión que nos permita actuar rápidamente al encontrarnos con un deslizamiento en las ruedas del monoplaza. Además de dar un valor preciso de la velocidad, tenemos que poder actualizar constantemente ese valor, es decir, disponer de un periodo de análisis lo suficientemente bajo como para que nuestro sistema detecte deslizamientos rápidamente y así pueda actuar en consecuencia.

Las variables físicas del monoplaza que entran en juego a la hora de medir la velocidad en las ruedas son las siguientes:

- Longitud de la circunferencia de la rueda: $1,65$ *metros*.
- Número de dientes del disco dentado acoplado a la rueda: 23 *dientes*.

En este punto, a grandes rasgos, podemos decir que existen *dos estrategias* para el cálculo de la velocidad de cada rueda; se expondrán las dos y se harán unos cálculos para comprobar la precisión que tenemos con cada una y elegir la más adecuada.

Estrategia 1 Una forma de medir la velocidad de una rueda, es contar los pulsos generados por el sensor en un determinado periodo de tiempo. De esta forma, comenzaríamos la medición de un tramo de tiempo fijado y cuando terminara empezaríamos la medición nuevamente, contando en cada medición el número de pulsos que el sensor ha generado. Con este sistema, la precisión de la medida es proporcional al periodo de muestreo. Si reducimos el periodo de muestreo, analizando así la velocidad más veces por segundo, la resolución en la medida se hace menor.

Si en un periodo de tiempo leemos 0 *pulsos*, determinaremos que la rueda en ese periodo iba a 0 *km/h*. El paso mínimo en la resolución de la medida es de 1 *pulso*, luego para conocer la resolución del sistema tenemos que conocer a qué velocidad equivale 1 *pulso*.

Para el cálculo de la velocidad usamos la fórmula (2).

$$Velocidad = \frac{DistanciaRecorrida}{TiempoEmpleado} \tag{2}$$

Donde la distancia, sería el avance de la rueda y para calcularlo basta con multiplicar el avance que efectúa la rueda entre dos pulsos consecutivos, por el número de pulsos generados, como vemos en la expresión (3). El avance de la rueda entre dos pulsos

consecutivos lo calcularemos dividiendo la longitud de la circunferencia de la rueda entre el número de dientes.

$$DistanciaRecorrida = (\frac{LongitudCircunferencia}{n^o dientes}) \cdot n^o pulsos \quad (3)$$

$$DistanciaRecorrida = (\frac{1{,}65}{23}) \cdot n^o pulsos$$

$$DistanciaRecorrida = 0{,}07174\ metros \cdot n^o pulsos$$

El paso mínimo en la resolución de la medida equivale a 1 *pulso*, así, por ejemplo si el periodo de muestreo es de 0,1 *segundos* (con esto analizaremos la velocidad de las ruedas 10 *veces por segundo*), con los datos de nº de dientes y longitud de la circunferencia, tenemos que 1 *pulso* detectado en ese periodo equivale a:

$$\frac{0{,}07174}{0{,}1} = 0{,}7174\ m/s = 2{,}58\ km/h$$

Con este periodo de análisis de 0,1 *segundos*, sólo obtendremos velocidades con incrementos de 2,58 *km/h*, como muestra la Tabla 4.

Tener esta resolución no es válido para nuestro sistema ya que a velocidades bajas introduce un error en la medida muy importante, que se acentúa cuando comparamos la velocidad de dos de las ruedas para hacer el cálculo del deslizamiento (ver apartado 6.5) y puede llegar a 2 * 2,58 = 5,16 *km/h*. A una velocidad del vehículo que ronde los 30 *km/h*, solo con este error de medida, el sistema puede detectar un deslizamiento de hasta un 17 %, algo intolerable para nuestro control de tracción.

Nº pulsos leídos en 0,1 *s.*	Velocidad de giro de la rueda
0 pulsos	0 *km/h*
1 pulso	2,58 *km/h*
2 pulsos	5,16 *km/h*
3 pulsos	7,74 *km/h*
4 pulsos	10,32 *km/h*
...	...

Tabla 4: Incremento en la velocidad en base al nº de pulsos leídos.

Si pretendemos bajar la resolución hasta obtener pasos de 0,5 *km/h* (0,138 *m/s*), deberemos ampliar el periodo de análisis. Haciendo los cálculos obtenemos lo siguiente:

$$\frac{1{,}65\ metros\ /\ 23\ dientes}{0{,}1388\ m/s} = 0{,}516\ segundos$$

Un análisis de la velocidad cada 0,516 *s.* (unas dos veces por segundo) haría que obtuviéramos esa resolución de 0,5 *km/h.* como muestra la Tabla 5.

Nº pulsos leídos en 0,516 *s.*	Velocidad de giro de la rueda
0 pulsos	0 *km/h*
1 pulso	0,5 *km/h*
2 pulsos	1 *km/h*
3 pulsos	1,5 *km/h*
4 pulsos	2 *km/h*
...	...

Tabla 5: Incremento en la velocidad en base al nº de pulsos leídos.

Para nuestro sistema de control de tracción, analizar dos veces por segundo la velocidad de las ruedas en busca de deslizamiento, es también insuficiente, pues no detectaremos rápidamente el momento en el que el coche empieza a perder adherencia a causa de una aceleración excesiva y no nos permitirá actuar lo antes posible. También ocurrirá que no detectaremos rápidamente cuándo el coche vuelve a recuperar la

adherencia haciendo que apliquemos la reducción de potencia durante más tiempo del que corresponde.

Esta forma de medir la velocidad la debemos desechar por no cumplir con los requisitos de nuestro sistema.

Estrategia 2 Otra forma de medir la velocidad de cada rueda es medir el tiempo que ha transcurrido entre dos pulsos consecutivos. Esta forma de medir la velocidad es mucho más precisa y nos permite tener un periodo de análisis muy bajo.

Nuevamente la fórmula para calcular la velocidad sería la reflejada en la expresión (2), anteriormente mostrada.

En este caso, es la distancia recorrida por la rueda, la que permanece fija y el tiempo empleado en recorrerla, es el que medimos cada vez. La distancia recorrida entre dos pulsos consecutivos es de:

$$DistanciaRecorrida = \frac{LongitudCircunferenciaRueda}{n^o dientes}$$

$$DistanciaRecorrida = \frac{1{,}65\ metros}{23\ dientes} = 0{,}07174\ metros$$

Depende del tiempo que transcurra entre los dos pulsos, tendremos una velocidad u otra. Para hacernos una idea de la magnitud de la señal que leemos del sensor (periodo y frecuencia) voy a mostrar en la tabla 6 estos valores a ciertas velocidades dadas, para una rueda.

Cada vez que un pulso es detectado, debemos haber medido el tiempo que ha transcurrido entre este pulso y su predecesor y entonces somos capaces de calcular

Velocidad *(km/h)*	0,5	2	5	10	40	120
VueltasRueda/seg	0,08	0,33	0,84	1,68	6,7	20,2
Pulsos/seg	1,93	7,74	19,6	38,7	154	464
Periodo *(ms)*	516	129	51,6	25,8	6,45	2,15

Tabla 6: Valores de la señal generada por el sensor a diferentes velocidades.

la velocidad de giro de la rueda. Vemos que cuando el coche circula a 10 *km/h*, los pulsos se suceden cada 25,8 *milisegundos* (38,7 *veces por segundo*) y van aumentando conforme aumenta la velocidad. Este sistema nos permite un periodo de análisis de la velocidad de cada rueda realmente bajo.

Vemos que cuando el coche circula a velocidad extremadamente baja, por ejemplo a 0,5 *km/h*, el sensor fijado en la rueda genera un pulso cada 0,516 *segundos*. Hasta después de 0,516 *segundos* no tendremos un nuevo pulso y por tanto, no podremos calcular la velocidad de la rueda, en este caso podríamos decir que el periodo de análisis es demasiado alto, pero es algo inherente al sistema, es una limitación física dado el número de dientes del disco dentado. A decir verdad, no es algo que deba preocuparnos puesto que nuestro objetivo de calcular rápidamente el deslizamiento sigue cumpliéndose ya que a esa velocidad es prácticamente imposible tener deslizamiento en una rueda.

Pensando en un caso extremo, con el vehículo parado y muy poca adherencia, al acelerar fuerte, las ruedas motrices derraparían y las ruedas delanteras marcarían 0 *km/h*. En este caso, al derrapar las ruedas traseras empezarían a leer velocidades bastante superiores a esos 0,5 *km/h*, posibilitando así una rápida detección del deslizamiento por parte del sistema, de hecho, en ese derrape o deslizamiento de las ruedas traseras, a partir de que la rueda gire más de 1/23 de vuelta (donde tendríamos ya un pulso por parte del sensor), ya estaremos detectando el deslizamiento.

6.4.2. Velocidad global del coche

Es importante tener una medida global y fiable de la velocidad del coche porque en nuestro sistema, dependiendo de la velocidad, vamos a ser más o menos "sensibles" a los deslizamientos producidos por el coche.

Como se hablaba en el apartado 5.1 de *Introducción al sistema empleado*, reduciremos la potencia del vehículo aplicando una señal a la centralita del monoplaza cuando exista deslizamiento. Pero para la generación de esa señal que reduzca la potencia, tenemos que tener en cuenta dejar un margen o umbral en el que permitamos un cierto deslizamiento de los neumáticos. Esto es así porque los neumáticos, aún encontrando algo de deslizamiento, siguen siendo efectivos proporcionándonos tracción y agarre. Se calcula que en buenas condiciones un neumático sigue siendo efectivo incluso trabajando con un 13 % de deslizamiento. Hablaremos más de este tema en el apartado 6.5 del *Cálculo del deslizamiento*.

En el sistema diseñado, ofrecemos la posibilidad de tener un umbral de deslizamiento permitido diferente para 3 franjas de velocidad. Con ello conseguimos más versatilidad en el setup del coche, haciendo, por ejemplo, más permisivo el sistema en salidas desde parado o a baja velocidad que circulando a altas velocidades, momento en el que podemos bajar ese umbral para sentir más confianza al acelerar fuerte en una curva.

En este momento, al introducir esta característica que acabamos de describir, se nos hace imprescindible conocer la velocidad actual del coche para aplicar el deslizamiento permitido correcto.

Nuestro sistema dispone de la velocidad de cada una de las 4 ruedas del coche en todo momento, pero no todas ellas son siempre válidas, para ello vamos a establecer los siguientes supuestos:

- Dado que el coche es tracción trasera, estando el coche totalmente parado, en una situación de baja adherencia de los neumáticos con el suelo, podemos encontrarnos que al acelerar fuerte, las ruedas traseras derrapen sin lograr generar ningún movimiento del coche. Este derrape también puede ocurrir al acelerar en una curva, así que desestimaremos la velocidad de las ruedas traseras para el cálculo de la velocidad global del coche.

- En las ruedas delanteras esto no va a ocurrir. Puesto que no son ruedas motrices, nunca girarán a más velocidad de la que realmente vaya el vehículo. Pero lo que sí suele ocurrir con frecuencia en competición es que al entrar en una curva algo pasado de velocidad, el piloto frene fuertemente a la vez que gire el volante, haciendo que las ruedas interiores, sobre todo la delantera, pierda muchísimo apoyo e incluso puede quedar al aire sin tocar el suelo. Al estar también accionando el freno, esta rueda se detendrá obteniendo así, en ese instante una lectura de velocidad igual a 0 *km/h*.

- Por último, al tomar una curva, las ruedas exteriores recorren mayor distancia que las ruedas interiores en un mismo espacio de tiempo, lo que se traduce como una mayor velocidad en estas ruedas. Este aspecto es un aspecto crítico a tener en cuenta a la hora de calcular el deslizamiento de la rueda, pero no tan crítico a la hora de determinar la velocidad global del coche. En una curva de baja velocidad, muy cerrada, de unos 10 *metros* de radio, la rueda interior presentaría un 10 % menos de velocidad que la exterior. Si la curva fuera de 20 *metros* de radio, la diferencia sería de tan sólo 5,4 %, lo que hace que para este aspecto en concreto, de la estimación de la velocidad global del coche, no lo tengamos en cuenta. En el apartado 6.5 del *Cálculo del deslizamiento* se mostrarán los cálculos para hallar la diferencia de velocidad de las ruedas para distintos radios de giro.

Después de lo expuesto, la estrategia para determinar la **velocidad global** del vehículo será establecer esta velocidad como la **velocidad de la rueda delantera que más rápido vaya**.

6.5. Cálculo del deslizamiento

En este apartado se describe la forma en la que se calcula el deslizamiento que presenta el coche basándonos en la velocidad leída en cada una de las ruedas.

¿A qué llamamos deslizamiento y cuándo una rueda presenta deslizamiento? El deslizamiento se produce cuando la velocidad de giro de la rueda, no se corresponde con la velocidad lineal del eje de ésta. Para entender estas palabras mejor, vamos a poner un ejemplo: Imagina que tenemos una rueda cuya longitud de su circunferencia es de 1 *metro*, entonces, al dar un giro completo, debemos avanzar 1 *metro* lineal. Si el avance ha sido menor, es que ha ocurrido deslizamiento por derrape o "sobregiro" (probablemente causado al acelerar bruscamente o simplemente al acelerar sobre firme resbaladizo) y también puede ocurrir que el avance haya sido mayor, por ejemplo en una frenada en la que ocurra un bloqueo de la rueda y el coche siga desplazándose.

¿Cómo lo calculamos? Siguiendo con el ejemplo anterior, una rueda cuya longitud de su circunferencia sea de 1 *metro* y haya girado una vuelta completa debe recorrer 1 *metro* de distancia. Si el avance ha sido, por ejemplo, de 0,8 *metros*, podemos calcular el deslizamiento como un porcentaje de la distancia que debería haber recorrido, tal y como muestra la fórmula 4. Pero si no se tienen presentes un par de consideraciones a la hora de hacer el cálculo podemos llegar a dar un valor que no tenga mucho sentido práctico. Por ejemplo, tomando la formula indicada anteriormente, si la velocidad lineal es 0, y la velocidad de giro es mayor de 0, la fórmula concurriría en una división por 0, cosa que nos dará error. Si nos movemos en velocidades muy cercanas a 0, el porcentaje de deslizamiento tendería a infinito, cosa que realmente no tendría mucho sentido práctico.

$$\%deslizamiento = \frac{(V_{giro} - V_{lineal})}{V_{giro}} \cdot 100 \tag{4}$$

$$\%\ deslizamiento = \frac{(1 - 0{,}8)}{1} \cdot 100 = 20\,\%$$

La idea, por tanto, es usar un porcentaje de deslizamiento acotado, en el que 0 % de deslizamiento signifique que no hay diferencia en las velocidades de giro y lineal de la rueda y 100 % signifique el máximo deslizamiento, es decir, que una de las velocidades (o de giro o lineal) fuera 0 y la otra un valor por encima de 0. Para esto, habría que calcular el deslizamiento siempre en referencia a la velocidad (o de giro o lineal) más alta de las dos, y el signo nos dirá cual de las dos es la que tiene mayor magnitud. Se usará por tanto la fórmula 5.

$$\%deslizamiento = \frac{(V_{giro} - V_{lineal})}{V_{mayor\ de\ entre\ las\ dos}} \cdot 100 \tag{5}$$

Si las dos velocidades son 0 no se puede aplicar la fórmula, porque existiría una división entre 0. En este caso, tendríamos el vehículo parado y se puede establecer directamente que el deslizamiento es 0 %. Haciendo el cálculo de esta manera, intuimos que si el deslizamiento es positivo, la rueda habrá presentado un "sobregiro" y si el deslizamiento es negativo, habrá ocurrido un bloqueo de la rueda.

Extrapolar esto a nuestro sistema no es algo inmediato, puesto que nosotros no podemos conocer el avance real de las ruedas, solo la velocidad de giro de ellas. ¿Cómo poder entonces calcular el deslizamiento? La solución empleada será la de comparar las velocidades de giro de las 4 ruedas del coche y analizándolas, obtener una medida fiable de deslizamiento.

El análisis global de las velocidades de giro de las ruedas conlleva hacer una serie de consideraciones y razonamientos que nos ayuden a determinar si existe deslizamiento. Estos razonamientos se expondrán en los siguientes puntos:

1. Realmente solo tenemos que centrarnos en los deslizamientos causados por aceleraciones, puesto que nuestro sistema actúa reduciendo la potencia del motor cuando existe deslizamiento y no corrigiendo la trayectoria como lo haría un sistema ESP (ver apartado 4.7).

2. Como ya hemos mencionado, en nuestro monoplaza, las ruedas traseras son las ruedas motrices y en situaciones de aceleración, no es posible que estas ruedas presenten menos velocidad de giro que las ruedas delanteras que no tienen ningún tipo de motricidad, ya que lo normal es que cuando se acelera fuertemente y se pierde adherencia, ocurra un sobregiro o derrape de las ruedas motrices (aumentando la velocidad de giro de la rueda en ese instante). Por esto concretamente, podemos tomar, en las ruedas delanteras, una medida más fiable de la velocidad del coche.

3. Continuando el razonamiento seguido en el anterior punto, podemos decir que los deslizamientos en las ruedas de nuestro monoplaza en los momentos de aceleración se presentarán en las ruedas traseras.

4. Una consideración muy importante que debemos tener en cuenta es que al tomar una curva las ruedas exteriores a la curva recorren más distancia que las interiores en un mismo periodo de tiempo, lo que da lugar a una mayor velocidad de giro de éstas. Este efecto se puede ver en la figura 39. Esta diferencia en la velocidad es dependiente del radio de giro de la curva y del ancho de vías del monoplaza. Se calculará para ver cómo afecta en el cálculo del deslizamiento.

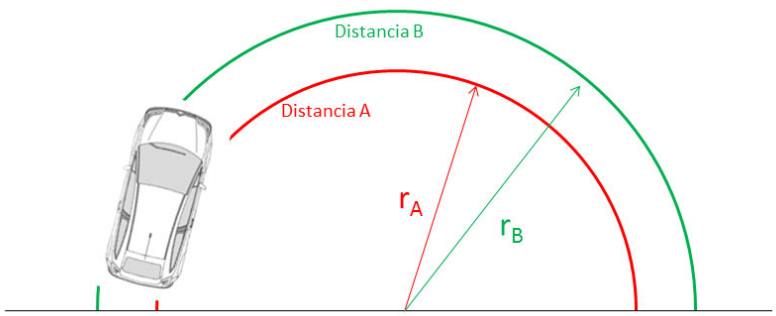


Figura 39: Vehículo tomando una curva

Teniendo en cuenta la figura 39, se pueden establecer las siguientes expresiones:

$$DistanciaA = \pi \cdot r_A$$

$$DistanciaB = \pi \cdot r_B$$

$$r_A = r_B - anchodevias$$

Hallaremos la diferencia, en porcentaje, entre las velocidades de giro de las ruedas interiores y exteriores en la curva. El dato que daremos es: en qué porcentaje es menor la velocidad de giro de la rueda interior respecto a la exterior, por lo que usaremos la expresión 6.

$$\%\ diferencia\ velocidad = \left(1 - \left(\frac{vel_{ruedainterior}}{vel_{ruedaexterior}}\right)\right) \cdot 100 \tag{6}$$

Desarrollamos uno de los términos en la expresión 6:

$$\left(\frac{vel_{ruedainterior}}{vel_{ruedaexterior}}\right) = \left(\frac{\frac{DistanciaA}{\cancel{tiempo}}}{\frac{DistanciaB}{\cancel{tiempo}}}\right) = \left(\frac{DistanciaA}{DistanciaB}\right) =$$
$$= \frac{\cancel{\pi} \cdot (r_B - anchodevias)}{\cancel{\pi} \cdot r_B} = \frac{r_B - anchodevias}{r_B}$$

Entonces, desarrollando la expresión 6, nos queda lo que podemos ver en la expresión 7

$$\%\ diferencia\ velocidad = \left(1 - \left(\frac{r_B - anchodevias}{r_B}\right)\right) \cdot 100 \tag{7}$$

El ancho de vías del monoplaza, que como se puede suponer, es la distancia entre el centro de las ruedas de un mismo eje (para facilitar los cálculos suponemos que el ancho delantero es igual al trasero, aunque en nuestro monoplaza hay 2 *cm* de diferencia), es de 1,15 metros. En la tabla 7 se muestran los diferentes porcentajes de diferencias de velocidad que presentan las ruedas en curvas con distintos radios de giro (para facilitar la muestra y comprensión de los datos, estableceremos el dato de "radio de giro" como el radio de la circunferencia que siguen las ruedas exteriores).

Radio de giro (metros)	Diferencia en %. Respecto a la vel. de la rueda exterior
5	23
10	11.5
15	7.67
20	5.75
25	4.6
30	3.83

Tabla 7: Tabla con datos

Viendo estos datos, podemos suponer que si no tenemos en cuenta este fenómeno y comparamos la velocidad de una rueda interior con una exterior en busca de deslizamiento, al tomar una curva, podemos encontrarnos con unas diferencias que nos pueden inducir a error ya que no evidencian, en absoluto, que haya habido tal deslizamiento.

Por poner un ejemplo, imaginemos nuestro monoplaza de Fórmula SAE haciendo la prueba del Skid PAD (ver *Figura 3* en el *Apartado 3.3.2*). En la prueba de Skid-Pad, se realizan unos giros con un radio aproximado de 9,5 metros, lo que generaría una velocidad de giro de 12,11 % menor en la rueda interior, no siendo esto un deslizamiento.

5. Otro punto a mencionar es que siendo estrictos, al comenzar a tomar una curva, tampoco las ruedas exteriores delantera y trasera hacen el mismo recorrido, como podemos ver en la figura 40. Pero en este caso, consideraremos despreciable la diferencia de velocidades entre las ruedas de un mismo lado del coche en el comienzo de un giro por ser muy pequeña y por tener un cálculo complejo.

Con todos estos razonamientos y consideraciones, el análisis del deslizamiento lo basaremos en **comparar las velocidades de giro de las ruedas de un mismo lado**, suponiendo que deben ir a la misma velocidad, y **hallando el porcentaje de diferencia** tendremos un cálculo estimado del deslizamiento.

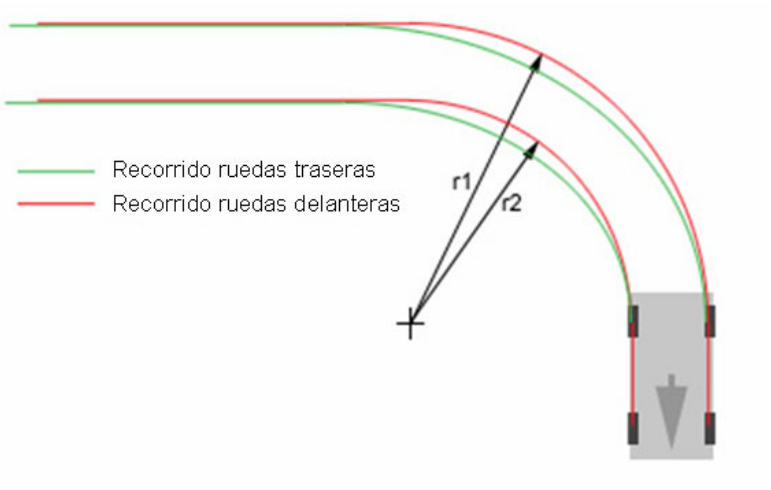


Figura 40: Recorrido de las ruedas en una curva.

Haciéndolo de esta forma, se calcularán dos deslizamientos, uno para las ruedas de una lado y otro para las del otro. En nuestro caso nos quedaremos con el mayor de los dos deslizamientos detectados a la hora de generar la señal de salida del sistema, para ser lo menos permisivos posibles con la aparición de este fenómeno.

El cálculo del deslizamiento de un lado, se basará en la fórmula 5 vista anteriormente, pero modificando las variables de tal forma que donde antes usabamos la V_{giro} ahora usaremos la *velocidad de la rueda trasera* y donde teníamos la V_{lineal} ahora pondremos la *velocidad de la rueda delantera*. La fórmula nos quedaría mostrada en la expresión 8.

$$\%deslizamiento\ en\ un\ lado = \frac{(V_{RuedaTrasera} - V_{RuedaDelantera})}{V_{mayor\ de\ entre\ las\ dos}} \cdot 100 \qquad (8)$$

Este porcentaje nos dirá cuánto menor es la velocidad más baja respecto a la más alta de entre las dos. Y así tendremos un porcentaje de deslizamiento comprendido entre 0 % y 100 %. En el rarísimo caso de que el valor fuera negativo, nos indicaría que la rueda trasera va más lenta que la delantera, lo que también sería un deslizamiento y habría que tenerlo en cuenta, de hecho, trabajaremos con el valor absoluto del deslizamiento calculado.

Como hemos adelantado, para el algoritmo del cálculo de deslizamiento se ha de tener presente que si el vehículo está parado, concurriríamos en una división por 0, así que es necesario detectarlo y establecer el deslizamiento a 0 % en esta situación. En este algoritmo también tendremos presente un problema añadido:

- Para valores muy cercanos a 0 *km/h* y también frente a errores de precisión en lectura de la velocidad cercana a 0 *km/h*, aplicando la fórmula nos puede dar valores de deslizamiento muy altos al poder tener diferencias porcentuales muy grandes entre los valores de velocidad leidos. Por ejemplo, si en un instante tenemos valores de velocidad de 1 *km/h* en una rueda y 0,1 *km/h* en otra, el deslizamiento sería del 90 %, algo totalmente excesivo para esas velocidades y que no reflejaría en absoluto una situación de derrapaje.

La solución será la de establecer un umbral en el que si una rueda va a una velocidad menor de 1 *km/h* se tome ese valor como velocidad de la rueda.

6.6. Bus CAN

El bus CAN (Controlled Area Network) es un protocolo de comunicaciones que hace uso de una topología tipo bus y sirve para el intercambio de información entre varios dispositivos conectados entre sí por medio del bus. Fue desarrollada por la firma alemana Robert Bosch GmbH en los años 80 específicamente para aplicaciones en el automóvil, aunque con el tiempo también se ha ido usando en el entorno industrial.

Se trata de un protocolo normalizado bajo un estándar ISO lo que garantiza la compatibilidad entre dispositivos fabricados por diferentes marcas.

Entre las ventajas que tiene podemos destacar:

1. Elimina cableado en los vehículos
2. Admite prioridad en los mensajes transmitidos
3. Sistema robusto
4. Detección de errores

6.6.1. Componentes del bus CAN

A grandes rasgos los componentes del bus CAN son:

1. Cable del bus: Se compone de un par de cables trenzados a los que los dispositivos se conectarán. Uno de los cables lleva la señal CAN_H y otro la señal CAN_L, el sistema es capaz de seguir funcionando si uno de los dos cables falla, trabajando con la otra señal con respecto a masa.
2. Resistencia de final de línea: Se conecta a ambos lados de la línea del bus para aumentar la calidad de la señal.
3. Nodos: Se conoce como nodo a cada uno de los dispositivos conectados al bus CAN.

Los nodos también se componen a su vez de los siguientes elementos:

1. Transceiver: Es el modulo que se conecta directamente a las líneas del bus, convierte y adecúa las señales y protege a los nodos frente a picos de tensión.
2. Controlador CAN: Es el módulo que recibe y envía los datos por el bus. El controlador CAN es el que hace que se sigan las reglas del protocolo en cuanto a temporización en el envío de los datos, prioridad de los mensajes, etc.

3. Unidad de procesado: El dispositivo en cuestión que se conecte al bus, debe tener una unidad centrar de proceso o CPU, que se encargue de entender y componer los mensajes a enviar/recibir en el bus.

En la Figura 41 se puede ver un esquema de la topología del bus CAN.

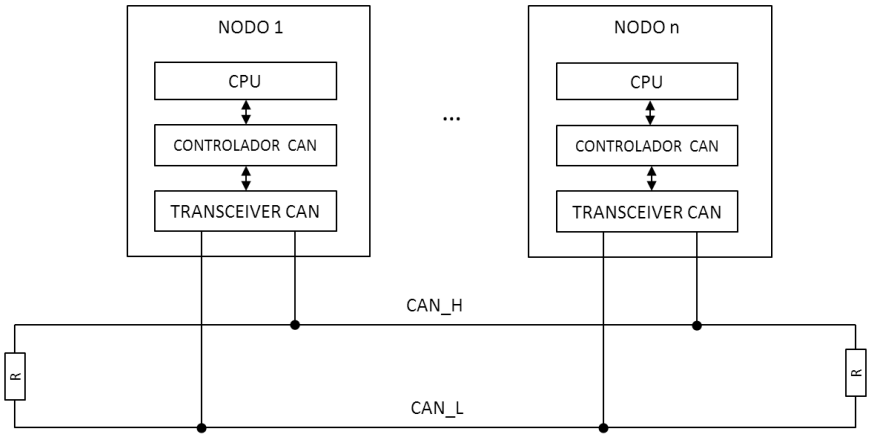


Figura 41: Topología del bus CAN.

En nuestro caso, el microcontrolador elegido, ya es en si mismo un pequeño computador que además, entre otros módulos, lleva integrado un controlador CAN.

6.6.2. Funcionamiento

El funcionamiento del protocolo CAN se basa en el envío y recepción de tramas, que son los mensajes que contienen los datos que se desean enviar y recibir. El bus es una red de difusión, en la cual, los mensajes enviados son recibidos por todos los nodos conectados al bus. Es cada nodo, el que tiene que decidir si guarda el mensaje o lo descarta. Cada trama tiene un identificador y con este identificador, un nodo en cuestión, sabe si la trama que ha recibido la debe guardar o no, dependiendo de si el dato que contiene es pertinente para dicho nodo o no lo es.

Imaginemos un dispositivo conectado al bus CAN, cuya función sea la de volcar información en el bus la temperatura que lee un sensor que dispone dicho dispositivo. El dispositivo constantemente (a una frecuencia determinada que dependerá de la relevancia que tenga el dato y de la velocidad del propio bus CAN), estará enviando una trama, con un identificador específico, que contiene el dato de la temperatura obtenida por el sensor. Todos los nodos conectados al bus, son capaces de guardar esa trama y así disponer de la temperatura si así lo necesitaran. Para ello, estos nodos tienen que conocer el identificador de la trama.

Vemos que los dispositivos conectados al bus CAN, deben de estar correctamente programados para enviar y recibir las tramas que cumplan la función para la cual fueron instalados en el sistema.

6.6.3. Tipos de tramas

Las tramas son agrupaciones de bits que se envían de forma consecutiva por el bus CAN y forman el mensaje correspondiente en código binario.

El protocolo CAN contempla 5 tipos de tramas:

- Trama de datos: Sirve para enviar información desde un nodo A a un nodo B.
- Trama de petición remota: Un nodo A envía una trama de petición remota cuanto desea solicitar a otro nodo el envío de un mensaje con un identificador concreto. En este caso la trama de petición remota lleva el mismo identificador de mensaje que el mensaje que se solicita.
- Trama de error: La transmite un nodo cuando detecta un error.
- Trama de sobrecarga: Sirve para proporcionar un retardo entre tramas sucesivas de datos o remotas.

- Espacio entre tramas: Cuando un nodo envía varias tramas seguidas siempre tiene que mandar un espacio entre tramas para que otros nodos puedan enviar tramas entre tanto y así no bloquear el bus.

Existen dos versiones en las especificaciones del protocolo CAN, que difieren básicamente en el tamaño del identificador de mensajes.

- Versión 2.0A: trabaja con identificadores de 11 bits

- Versión 2.0B: Esta versión permite trabajar con tramas que tenga identificadores de 29 bits, para ello, la trama de datos se divide en trama de datos estándar (para mensajes con identificador de 11 bits) y trama de datos extendida (trabaja con identificadores de mensaje de 29 bits). A su vez, la versión 2.0B tiene dos subversiones:

  1. Versión 2.0B pasivo: ignora los mensajes con identificadores de 29 bits
  2. Versión 2.0B activo: maneja identificadores de 11bits y de 29 bits.

El formato de una trama de datos en la versión 2.0B activo se puede ver la Figura 42

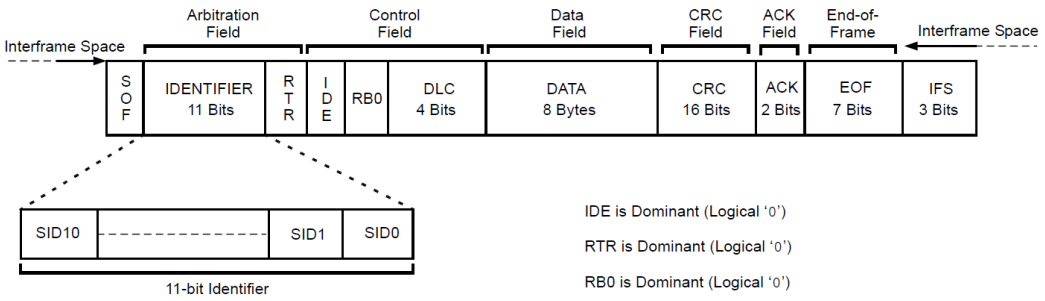


Figura 42: Formato de una trama de datos estándar.

6.6.4. Uso de las tramas en el sistema de control de tracción

Nuestro sistema usará el protocolo bus CAN para poder configurar ciertos parámetros y para enviar información acerca de su estatus y configuración actual.

El microcontrolador que se va a utilizar lleva incorporado un controlador CAN que trabaja en el estándar 2.0B activo. Solo usaremos tramas de datos estándar. El campo "DATA" es de 8 bytes y lo dividiremos en 8 datos de 1 byte para estructurar el contenido y facilitar la labor de procesado del mensaje.

Tramas enviadas

Solo enviaremos tramas de datos cuando desde otro nodo del bus nos lo pidan por medio de una trama de petición remota. Por tanto, configuraremos el controlador CAN para que responda a estas tramas de petición remota. Las tramas que nosotros enviaremos contendrán datos de estatus y valor actual de los parámetros del sistema. A estas tramas las llamaremos "mensajes de estatus" y serán las siguientes:

- Mensaje STATUS0:

  Identificador de mensaje: 0x7F0

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
DIENTES	%DAV	%DMV	%DBV	CTPE	CTPD	CDPE	CDPD

Tabla 8: Contenido del campo data

  - Byte 7: número de dientes del disco dentado. Valores posibles de 0 a 255 (unsigned char).
  - Byte 6: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).
  - Byte 5: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).

- Byte 4: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).
- Byte 3: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
- Byte 2: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).
- Byte 1: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
- Byte 0: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).
- NOTA: Si el valor de la variable que guarda el dato de la longitud de la circunferencia de una de las ruedas es de 165'48 cm, entonces en el byte correspondiente a la parte entera estará almacenado el valor 165 y en el byte correspondiente a la parte decimal, el valor 48. No podemos guardar valores con 3 decimales, ni longitudes superiores a 255,99 cm.

- Mensaje STATUS1:

  Identificador de mensaje: 0x7F1

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
UAV	UBV	TiA1	TiM1	TiB1	KpA1	KpM1	KpB1

Tabla 9: Contenido del campo data

- Byte 7: Umbral alta velocidad. Valor, en Km/h, que delimita cuando vamos a "alta velocidad" y cuando vamos a "media velocidad". Es solo un convenio para la regulación que hace el sistema. Valores posibles de 0 a 255 (unsigned char).
- Byte 6: Umbral baja velocidad. Valor, en Km/h, que delimita cuando vamos a "media velocidad" y cuando vamos a "baja velocidad". Es solo un convenio para la regulación que hace el sistema. Valores posibles de 0 a 255 (unsigned char).

- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 1: Constante proporcional a media velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

- Mensaje STATUS2:

  Identificador de mensaje: 0x7F2

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
-	-	TiA2	TiM2	TiB2	KpA2	KpM2	KpB2

Tabla 10: Contenido del campo data

- Byte 7: No se usa.
- Byte 6: No se usa
- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 1: Constante proporcional a media velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de Çonstante Proporcionalz "Tiempo Integral"serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión

alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

- Mensaje STATUS3:

  Identificador de mensaje: 0x7F3

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
-	-	TiA3	TiM3	TiB3	KpA3	KpM3	KpB3

Tabla 11: Contenido del campo data

- Byte 7: No se usa.
- Byte 6: No se usa
- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).

- Byte 1: Constante proporcional a media velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

Tramas recibidas

Cuando el sistema reciba una trama de datos con un identificador estándar determinado, el sistema lo guardará y procesará el dato contenido en la trama para actualizar las variables correspondientes en el programa, así el usuario podrá cambiar el valor de los parámetros del sistema para un ajuste fino. El sistema usa 5 identificadores de mensaje diferentes para contener todos los parámetros modificables. A estas tramas las llamaremos "mensajes de configuración" y serán los siguientes:

- Mensaje CONF0:

  Identificador de mensaje: 0x7F4

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	-	DIENTES	CTPE	CTPD	CDPE	CDPD

Tabla 12: Contenido del campo data

- Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0, nuestro sistema no procese los datos y los almacene. Para ello, antes comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.
- Byte 6: No se usa.
- Byte 5: No se usa.
- Byte 4: número de dientes del disco dentado. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
- Byte 2: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).
- Byte 1: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
- Byte 0: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).
- NOTA: Si el valor de la variable que guarda el dato de la longitud de la circunferencia de una de las ruedas es de 165,48 *cm*, entonces en el byte correspondiente a la parte entera estará almacenado el valor 165 y en el byte correspondiente a la parte decimal, el valor 48. No podemos guardar valores con 3 decimales, ni longitudes superiores a 255,99 *cm*.

▪ Mensaje CONF1:

Identificador de mensaje: 0x7F5

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	-	UAV	UBV	%DAV	%DMV	%DBV

Tabla 13: Contenido del campo data

- Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0, nuestro sistema no procese los datos y los almacene. Para ello, antes comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.
- Byte 6: No se usa.
- Byte 5: No se usa.
- Byte 4: Umbral alta velocidad. Valor, en Km/h, que delimita cuando vamos a "alta velocidad" y cuando vamos a "media velocidad". Es solo un convenio para la regulación que hace el sistema. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Umbral baja velocidad. Valor, en Km/h, que delimita cuando vamos a "media velocidad" y cuando vamos a "baja velocidad". Es solo un convenio para la regulación que hace el sistema. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).
- Byte 1: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).
- Byte 0: Porcentaje de deslizamiento admitido a alta velocidad. Valores posibles de 0 a 100 (unsigned char).

▪ Mensaje CONF2:

Identificador de mensaje: 0x7F6

- Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0,

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	TiA1	TiM1	TiB1	KpA1	KpM1	KpB1

Tabla 14: Contenido del campo data

nuestro sistema no procese los datos y los almacene. Para ello, antes comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.

- Byte 6: No se usa.
- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 1: Constante proporcional a media velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 1 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de “Constante Proporcional” y “Tiempo Integral” serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados

por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

- Mensaje CONF3:

  Identificador de mensaje: 0x7F7

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	TiA2	TiM2	TiB2	KpA2	KpM2	KpB2

Tabla 15: Contenido del campo data

- Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0, nuestro sistema no procese los datos y los almacene. Para ello, antes comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.
- Byte 6: No se usa.
- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).

- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 1: Constante proporcional a media velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 2 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

- Mensaje CONF4:

  Identificador de mensaje: 0x7F8

  - Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0, nuestro sistema no procese los datos y los almacene. Para ello, antes

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	TiA3	TiM3	TiB3	KpA3	KpM3	KpB3

Tabla 16: Contenido del campo data

comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.

- Byte 6: No se usa.
- Byte 5: Constante de tiempo integral a alta velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 4: Constante de tiempo integral a media velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 3: Constante de tiempo integral a baja velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 2: Constante proporcional a alta velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 1: Constante proporcional a media velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- Byte 0: Constante proporcional a baja velocidad para el mapa 3 del regulador del control de tracción. Valores posibles de 0 a 255 (unsigned char).
- IMPORTANTE: todos los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados multiplicados por 10. En el programa principal, estos valores también se almacenan multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que

hacer conversión alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 0,1. Si guardamos el valor 123 en el byte correspondiente a KpB1, el valor real de la constante proporcional a baja velocidad del mapa 1 será 12,3. Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores tomados por las constantes proporcionales y los tiempos integrales del regulador comprenderán el rango de 0 a 25,5 y siempre con incrementos de 0,1.

En el apartado 10 se explicará el proceso de modificación de los parámetros del sistema mediante los mensajes CAN.

6.7. PWM: cálculo de la frecuencia y ciclo de trabajo

La modulación PWM (de las siglas en inglés Pulse-width modulation) es una técnica en la que se modifica el ciclo de trabajo de una señal periódica para enviar información o controlar la cantidad de energía que se envía.

El ciclo de trabajo de una señal periódica es la relación que existe entre el tiempo que se mantiene la señal en estado activo (o a nivel alto) y el tiempo que se mantiene a nivel bajo en un mismo periodo de la señal.

Nuestro sistema generará una señal cuadrada con ciclo de trabajo variable que será proporcional al porcentaje de deslizamiento detectado en las ruedas. Esta señal se la mandamos a la centralita del vehículo. Cuando esta señal está a nivel alto, la centralita anula la descarga de las bujías de los cilindros y no se produce chispa, por lo que no existirá combustión y por tanto no se generará potencia en ese tiempo de

combustión. Cuanto la señal enviada está a nivel bajo, la centralita hará que las bujías funcionen con normalidad.

En este punto, nos encontramos con la tesitura de establecer los valores óptimos de frecuencia y de rango de ciclo de trabajo de la señal, para proporcionar una reducción de potencia justa en el motor y que así presente un funcionamiento suave cuando el sistema entre en acción.

Si el sistema, en situación de salida desde parado del monoplaza, genera una señal a nivel activo durante demasiado tiempo, quitaremos toda la potencia del motor, el coche detendrá su avance y es posible que si vamos a baja velocidad, el coche se termine calando.

Para solventar esto es necesario realizar un estudio de las revoluciones a las que trabaja el motor y establecer la temporización de los tiempos de admisión, compresión, combustión y escape en los cilindros.

En el motor, al ser de 4 cilindros y 4 tiempos, desde el punto muerto superior al punto muerto inferior, hay media vuelta del cigüeñal. En esta media vuelta, cada uno de los cilindros está en un tiempo diferente de los 4 posibles: admisión, compresión, combustión y escape. Esto hace que en media vuelta del cigüeñal se produzca una chispa de bujía en uno de los cilindros. A la siguiente media vuelta, se producirá la chispa en otro cilindro y así hasta completar los 4 cilindros y comenzar nuevamente el ciclo. En las Figuras 43 y 44 podemos ver la temporización de los tiempos del motor:

Por tanto, 1 revolución o vuelta del cigüeñal, equivale a 2 tiempos del motor. Calculando la duración de 1 solo tiempo del motor, tendremos el periodo de la generación de la chispa.

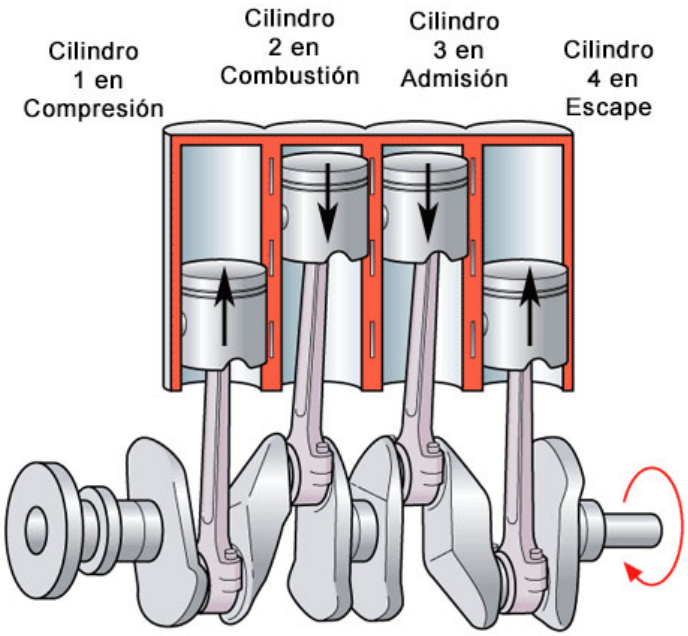


Figura 43: Tiempos del motor.

El motor que usa el monoplaza del equipo UMPRacing, puede llegar a las 11000 rpm, pero la posibilidad de deslizamiento en las ruedas por una entrega excesiva de potencia la tendremos por encima de las 3000 rpm. Calculamos el periodo de chispa:

- A 3000 rpm:

$$\frac{3000\ rpm}{60\ seg/min} \cdot 2\ tiempos/rev = 66{,}66\ tiempos$$

$$T_{chispa} = \frac{1}{66{,}66} = 10\ milisegundos$$

- A 11000 rpm:

$$T_{chispa} = 2{,}72\ milisegundos$$

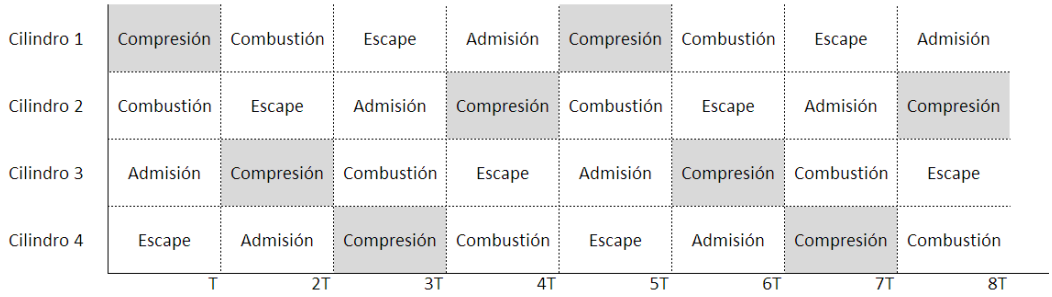


Figura 44: Tiempos del motor.

Para la elección de la frecuencia de la señal PWM nos basaremos en el periodo de chispa a 3000 rpm ya que una frecuencia menor de la señal PWM siempre contendrá los periodos de chispa a mayores rpm, pero al revés puede no sería así.

En la Figura 45 se muestra un gráfico con los periodos de chispa del motor en cada cilindro a 3000 rpm. Un asterisco significa que en ese instante se produce una chispa en el cilindro señalado.

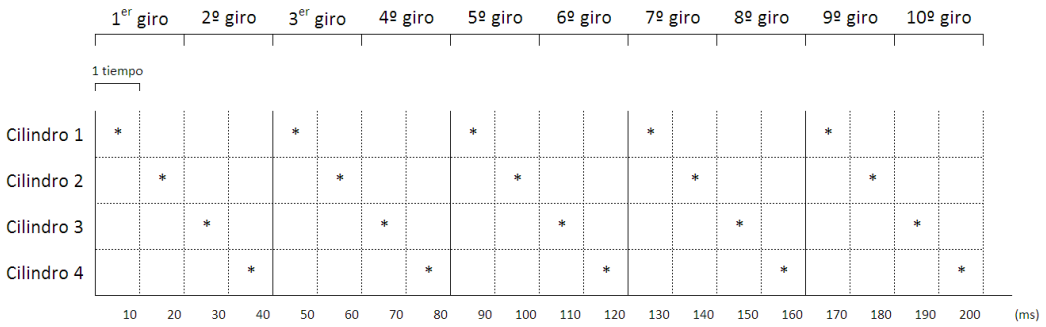


Figura 45: Descargas de las bujías a 3000 rpm.

Para la elección de la frecuencia y ciclo de trabajo de la señal cuadrada de salida, estableceremos que vamos a suprimir, al menos, una chispa cada 10 chispas que se produzcan a 3000 rpm. Esto nos da una señal de una frecuencia de 10 Hz y un ciclo de trabajo de 10 %. En la Figura 46 podemos ver cómo queda la señal superpuesta al gráfico anteriormente mostrado. Estableceremos este criterio porque la duración mínima del nivel activo de la señal cuadrada de salida debe ser igual al tiempo de compresión a 3000 rpm (si es más pequeñá no estaremos seguros de suprimir siempre

al menos una chispa y si es mayor podemos suprimir más de una chispa seguida, cosa que no queremos). Por tanto, la duración mínima del nivel activo de la señal cuadrada debe ser 10ms.

Por otro lado, si queremos un periodo de análisis del sistema lo suficientemente alto, no podemos disminuir mucho la frecuencia de la señal. Estableciendo la frecuencia a 10 Hz (10 veces por segundo), tenemos que el ciclo de trabajo mínimo será del 10 %. Mayor frecuencia haría que el ciclo de trabajo aumentara y perderíamos mucho en progresividad a la hora de restar potencia al motor.

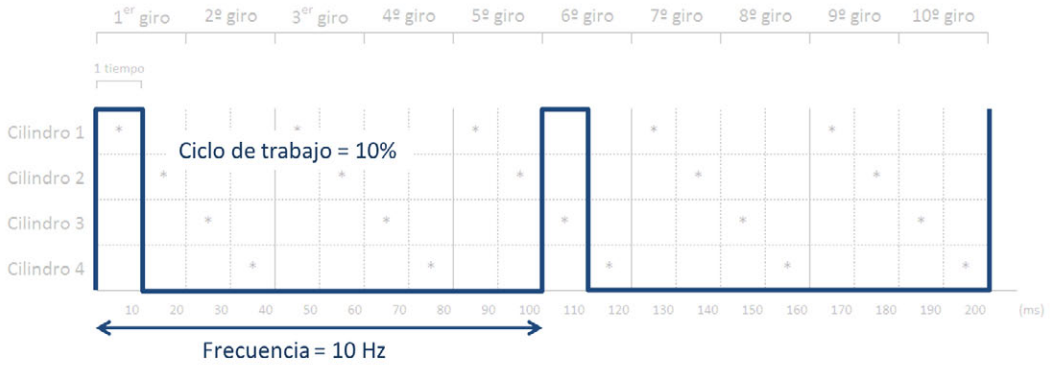


Figura 46: Señal cuadrada de salida del sistema.

Al aplicar un ciclo de trabajo del 10 %, el motor debería ver reducida su potencia en ese porcentaje también. Esta será el mínimo ciclo de trabajo aplicable. Vemos que si debiéramos aumentar el ciclo de trabajo de la señal, para conseguir suprimir más chispas, tenemos que hacerlo en cantidades del 10 %. Así pues, 10 % será el paso mínimo del ciclo de trabajo, pudiendo generar señales con 10 %, 20 %, 30 %, etc. de ciclo de trabajo.

Como no podemos hacer pruebas reales del funcionamiento en el coche de este método de reducción de potencia, seremos precavidos y limitaremos el ciclo de trabajo máximo al 30 % porque es posible que en el arranque del coche, si suprimimos más de 3 chispas seguidas el motor pueda calarse.

6.8. Parámetros del regulador del control de tracción

El software del dispositivo implementa un regulador PI para el control del deslizamiento en las ruedas del vehículo. La acción más importante del controlador será la acción proporcional, que generará una señal de control proporcional al tanto por ciento de deslizamiento detectado, pero le añadiremos la acción integral porque en situaciones de muy baja adherencia de las ruedas, es posible que el coche tarde más tiempo en recuperar el control y será interesante que en este caso exista un aumento de la señal de control proporcional a la duración del error y al valor de este.

En este apartado elegiremos los valores de las constantes proporcional y tiempo integral y tendremos en cuenta aspectos particulares de la dinámica de nuestra sistema, que hará que tengamos que añadir ciertos algoritmos para su control.

El diagrama de bloques del regulador implementado por el dispositivo se ve reflejado en la figura 47

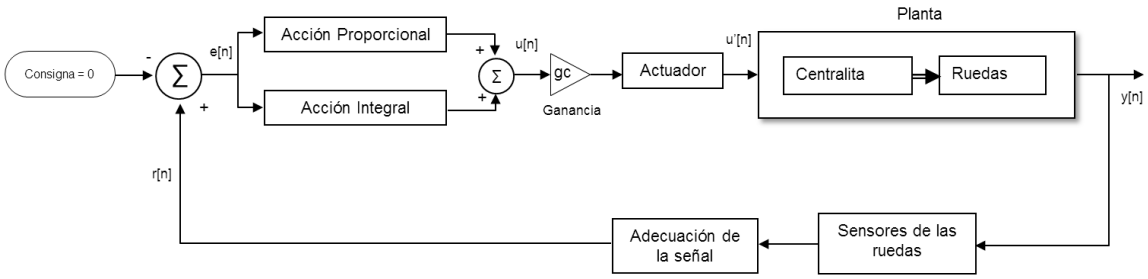


Figura 47: Diagrama de bloques del regulador PI.

Al no poder hacer pruebas reales del método de disminución de potencia del motor, haremos estimaciones a la hora de elegir los valores de constante proporcional y tiempo integral en base a la señal que queramos enviar a la centralita cuando encontremos deslizamiento el sistema.

Es importante tener en cuenta, que vamos a implementar la posibilidad de elegir entre 3 modos de actuación del sistema de control de tracción. Además, dependiendo de la velocidad del coche, el regulador actuará de forma diferente, teniendo 3 franjas de velocidad: vehículo a baja velocidad, a media velocidad y a alta velocidad. Todo esto hará que tengamos diferentes valores de constante proporcional y tiempo integral para cada franja de velocidad y para cada modo de actuación.

¿Por qué diferenciamos entre franjas de velocidades? La respuesta del control de tracción no puede ser la misma a todas las velocidades porque el comportamiento del coche tampoco lo es. A altas velocidades, existe menor probabilidad de que se genere un deslizamiento ocasionado por excesiva potencia del motor, pero de encontrarnos con él, por ejemplo a causa de un mal estado del firme, debemos reducir la potencia más rápidamente porque nos encontramos en una situación más peligrosa. En cambio, en salidas desde parado, o circulando a baja velocidad, podríamos querer que el sistema se hiciera más permisivo al encontrar deslizamiento, por ello, se da la opción de tener 3 franjas de velocidad diferentes.

6.8.1. Valores de la señal de error

La señal de error, será el dato del deslizamiento calculado de la forma que se indicó en el apartado 6.5 y dado en tanto por 1. Esto quiere decir que tendremos un error que va desde 0 (cuanto el sistema no presenta deslizamiento) hasta 1, que es el máximo deslizamiento (y que representa que una rueda esté parada mientras otra esté en movimiento).

Cuando hablamos del cálculo del deslizamiento en el apartado anteriormente mencionado, también comentamos la necesidad de permitir un porcentaje de deslizamiento, ya que los neumáticos aun trabajando con un leve deslizamiento (en torno a un 10 %, dependiendo del material, temperatura y terreno de uso) son capaces de traccionar de forma efectiva.

Para incluir este aspecto en el sistema, incorporaremos un algoritmo en la función del cálculo del deslizamiento, que haga que cuando el deslizamiento esté en una franja de valores concreta, lo convierta en 0.

También ocurre que a bajas velocidades nos encontraremos con errores mayores que a altas velocidades porque un derrape circulando a baja velocidad, ocasiona un deslizamiento mayor que a altas velocidades, debido a que porcentualmente las diferencias de velocidad de las ruedas son mayores.

Para solucionar esto último, daremos la opción de configurar diferentes valores de deslizamiento permitido cuando nos encontremos a baja velocidad, media velocidad y alta velocidad. Con esto tendremos un sistema más flexible y configurable.

6.8.2. Señal de control

Ya vimos en el apartado 6.7 que la señal que tiene que generar el sistema será una señal cuadrada de 10Hz de frecuencia y ciclo de trabajo variable. En este caso el ciclo de trabajo, será la variable manipulada. Dependiendo del % de deslizamiento detectado, o más concretamente, de la señal de control calculada por el regulador, cambiaremos el valor del ciclo de trabajo de la señal que el sistema enviará a la centralita.

La señal de control, denotará el tanto por cierto de ciclo de trabajo que debe tener la señal de salida.

No hay que olvidar que, como decíamos en el apartado 6.7 vamos, a restringir la señal de salida de dos formas:

1. Restringiendo el valor máximo de ciclo de trabajo a un 30 %
2. Limitando el ciclo de trabajo a incrementos del 10 %.

Estas dos cosas las haremos a través del actuador, que será una función en nuestro software.

A través del actuador, generaremos la señal física que saldrá de nuestro dispositivo y aplicaremos un algoritmo que hará que se cumplan las limitaciones anteriormente citadas.

6.8.3. Algoritmo de reset y saturación

Cuando explicábamos las técnicas de anti saturación de la acción integral o Wind-up, veíamos la necesidad de incorporar alguna de ellas a nuestro sistema.

Si el coche presenta un deslizamiento durante un tiempo prolongado, la acción integral del regulador se irá incrementando hasta el momento que el deslizamiento desaparezca. Pero el valor de la acción integral permanecerá aun cuando no exista error en el sistema y no se reducirá hasta que no haya error negativo que haga disminuir su valor. En nuestro sistema no existe error negativo, entonces tenemos que idear una forma que haga disminuir su valor a cero. Además, para mejorar el funcionamiento del sistema, dosificaremos la entrega de potencia del motor cuando la velocidad del monoplaza sea baja, para ello, continuaremos aplicando cierto valor a la señal de control durante un tiempo definido aún después de que el error se haya hecho cero.

¿Por qué dosificaremos la entrega de potencia del motor? Esto sobre todo es muy interesante en las salidas desde parado y en condiciones de muy poca adherencia. En el momento en el que el coche acelera y se pone en marcha, el motor entrega mucha potencia y tener deslizamiento es muy probable. En este caso cuando detectamos deslizamiento, disminuimos la potencia del motor para solventarlo y volver a tener el deslizamiento controlado. Si en este punto, devolvemos toda la potencia al motor, es muy probable que se vuelva a presentar deslizamiento ya que el piloto, en estas condiciones de salida, mantendrá grandes dosis de aceleración.

Para tal efecto, se ha pensado un algoritmo que restablezca el valor de la acción integral e incluya además la parte de la dosificación en la entrega de potencia del motor. Este algoritmo actuará poniendo a cero la acción integral en cuanto deje de existir deslizamiento (o esté dentro de la franja de deslizamiento admitido) y además hará que se continúe ejerciendo una señal de control de valor 0,1 (lo que equivale a un 10 % de ciclo de trabajo) durante 500 *milisegundos*, solo cuando estemos circulando a baja velocidad.

Necesitaremos saber el periodo de muestreo del sistema para seguir aplicando la señal de control durante esos 500 milisegundos, lo que nos dará durante cuantas iteraciones del sistema la señal de control debe seguir actuando.

Periodo de muestreo

Para el funcionamiento del control de tracción se ha establecido que el sistema analice la velocidad de las ruedas 10 veces por segundo y actúe en consecuencia. Por lo que el periodo de muestreo del sistema será de 100 ms.

6.8.4. Cálculo de las constantes

Como comentamos anteriormente, al no poder hacer pruebas reales del método de disminución de potencia del motor, haremos estimaciones a la hora de elegir los valores de constante proporcional y tiempo integral en base al ciclo de trabajo de la señal cuadrada que queremos enviar a la centralita.

El error del sistema es el deslizamiento detectado en tanto por uno. La señal de control indicará el porcentaje de ciclo de trabajo de la señal de salida, pero lo hará también en tanto por 1, luego el valor de saturación de la señal de control será de 1, que significará un 100 % en el ciclo de trabajo de la señal de salida.

Con estas premisas, elegiremos un valor de constante proporcional más bien comedido que por cada 10 % de deslizamiento encontrado, genere un 8 % de ciclo de trabajo de la señal de salida:

$$Kp = \frac{0{,}08}{0{,}1} = 0{,}8$$

Para la constante de tiempo integral, seremos aún más reservados, y en principio la ajustaremos para que aporte a la señal de control poco incremento. Estimaremos que vamos a querer un incremento de la señal de control del 15 % después de encontrarnos con un deslizamiento del 10 % que dure 1 segundo. Los cálculos serían los siguientes:

$PeriodoMuestreo = 100\ ms$, entonces $1\ segundo = 10\ periodosdemuestreo$

$$0{,}15 = \frac{1}{Ti} \cdot (0{,}1 \cdot 10)$$

$$Ti = 6{,}66$$

Recordar que tenemos 3 modos de selección de control de tracción y que a su vez, cada modo, dispone de 3 constantes proporcionales y tres constantes de tiempo integral, una por cada franja de velocidad estipulada.

Será necesario hacer pruebas en el vehículo para calibrar y ajustar los valores de las constantes evaluando la reducción del deslizamiento en base a la reducción de potencia del motor.

6.9. Diagrama de bloques y señales involucradas en el sistema

El sistema se compondrá de una placa de circuito impreso principal, que albergará el microcontrolador, la regulación de tensión de alimentación y en general todos los circuitos necesarios para su funcionamiento.

También contaremos con un mando de control remoto para que el piloto pueda elegir entre los diferentes modos de actuación del control de tracción.

Estos modos serán los 4 siguientes:

1. Control de tracción desactivado
2. Control de tracción activado en el mapa 1
3. Control de tracción activado en el mapa 2
4. Control de tracción activado en el mapa 3

Cada mapa del control de tracción se puede configurar para actuar con diferente intensidad ante la aparición de deslizamiento en el vehículo.

El mando de control remoto contará con los siguientes elementos:

- Mando selector de 4 posiciones
- Cuatro leds que indicarán la selección de cada uno de los modos elegidos.
- Un led que indicará que la PCB recibe alimentación

La PCB tendrá que ser recubierta con algún tipo de carcasa y dotada de elementos de fijación necesarios para colocarlo en el dashboard del monoplaza o en el sitio que determine el equipo UPMRacing.

Entrando en detalle de las partes que compondrán la PCB y las señales involucradas en ella, en la Figura 48 se puede ver un diagrama de bloques de la composición del sistema y en el siguiente listado desglosaremos las señales de entrada y salida del sistema.

- POWER: Alimentación del sistema que se proveerá desde la batería del vehículo. Llegará a través de dos cables (GND y +12V) y se unirán a la PCB mediante un conector de dos vías.

- Sensores de efecto Hall: Desde los sensores de efecto Hall de las ruedas vendrán 4 cables (uno por sensor) que transportan la señal de éstos y se conectan a la PCB por medio de dos conectores de dos vías cada uno. La salida de los sensores de efecto Hall es de colector abierto, lo que deriva en que los estados de la señal serán GND o bien alta impedancia. Utilizaremos solo un cable para cada sensor, dando por hecho que la señal de masa es común en todo el vehículo.

- Salida hacia la centralita. Usaremos una entrada digital de la centralita PE-ECU-1 del vehículo. Mediante esta entrada, la ECU atiende señales de 0V a 5V. Esta señal irá desde nuestros dispositivo de control de tracción hasta la centralita por medio de dos cables (GND y positivo de la señal) mediante un conector de dos vías. Si no hiciera falta conectar la señal GND, por tratarse de una señal común en todo el vehículo, solo se tirará un cable desde el conector de dos vías de la PCB.

- Conector bus CAN: El conector bus CAN es un conector DB9, comúnmente utilizado para interconectar nodos en el bus CAN.

- Conector RJ11 para la programación del dispositivo. Este conector llevará un cable desde el programador MPLAB ICD2 hasta nuestra PCB y sirve para realizar las tareas de programación y depurado de código en tiempo real.

- Conector RJ45. El sistema dispone de un control remoto para que el piloto pueda cambiar los modos de funcionamiento del control de tracción y también apagarlo

si lo desea. La conexión de la PCB principal y este control remoto se realiza a través de un cable de 8 hilos con conectores RJ45. El pineado de este conector es el siguiente:

- Pin1: Señal Controlbit0. Junto con la señal del pin 2 (Controlbit1) servirá para indicar al microcontrolador qué modo de funcionamiento ha sido seleccionado.
- Pin2: Señal Controlbit1
- Pin3: Señal de encendido del led "tcsoff" que indica que el sistema está desactivado
- Pin4: Señal de encendido del led "mapa1" que indica que el mapa1 está seleccionado
- Pin5: Señal de encencido del led "mapa2" que indica que el mapa2 está seleccionado
- Pin6: Señal de encendido del led "mapa3" que indica que el mapa3 está seleccionado
- Pin7: VCC, + 5 voltios
- Pin8: GND, 0 voltios.

En el apartado 8.2 se podrán ver los esquemáticos de los circuitos de la PCB para tener ya un conocimiento completo del diseño de ésta.

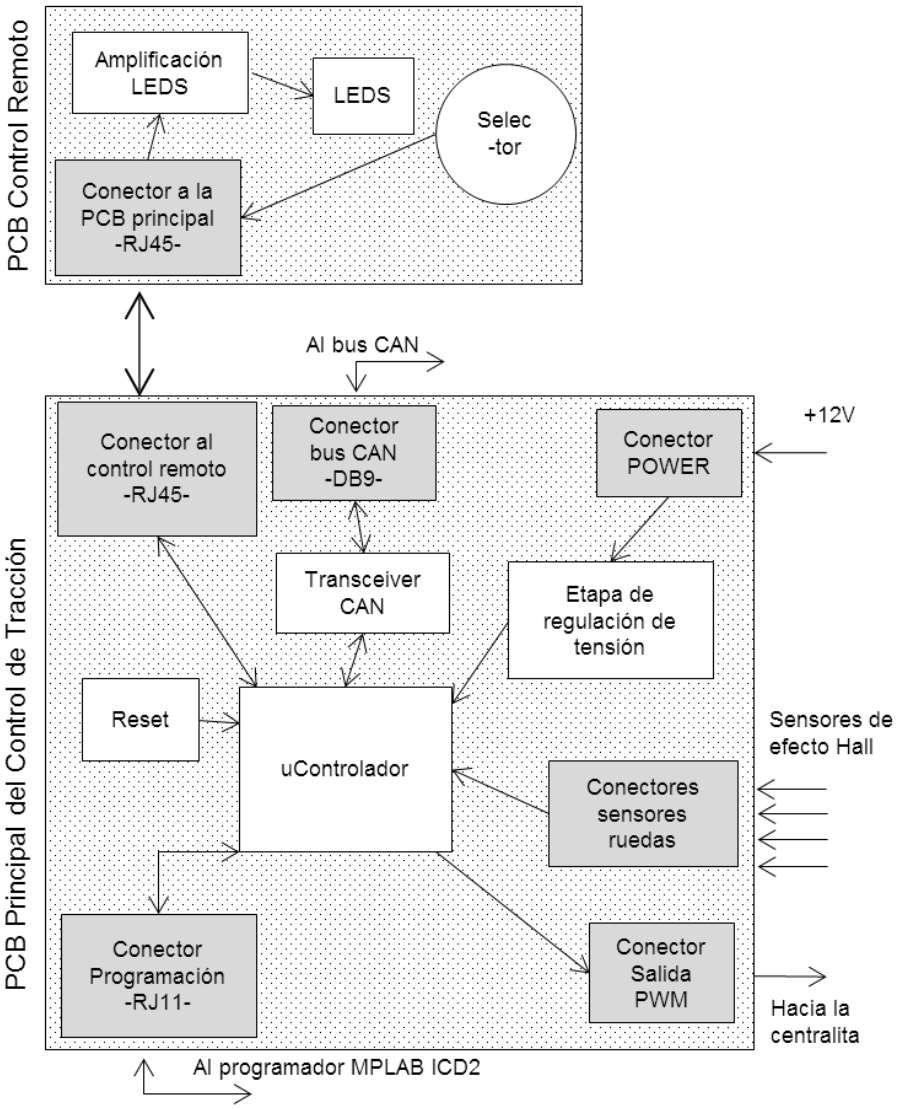


Figura 48: Diagrama de bloques y señales.

7. El programa

7.1. Entorno de desarrollo

El entorno de desarrollo para la programación del microcontrolador será el software de Microchip MPLAB IDE en su versión 8.76.

El entorno de desarrollo es un conjunto completo de utilidades para abordar la elaboración de un programa. Incluye un editor de código, un compilador, un simulador y otras herramientas. Junto con la instalación del programa MPLAB IDE, se instalan librerías para el uso de los diferentes microcontroladores de la firma Microchip.

El programa embebido en el microcontrolador lo escribiremos en lenguaje C. Seguidamente se presentarán los diagramas de flujo y las funciones más importantes usadas en el código generado.

7.2. Diagramas de flujo y funciones

Se presentarán los diagramas de flujo desde el más general al más particular. También se detallarán las cabeceras de las funciones más importantes así como sus parámetros de entrada y salida.

Los diagramas de flujo y funciones descritos en este libro serán las siguientes:

1. Diagrama de flujo general del funcionamiento del sistema.
2. Diagrama de flujo de la función main.
3. Función del cálculo de deslizamiento y su diagrama de flujo

4. Función del controlador y su diagrama de flujo

5. Función del actuador y su diagrama de flujo

El diagrama de flujo general y el de la función “main” se pueden ver en las Figuras 49 y 50 respectivamente.

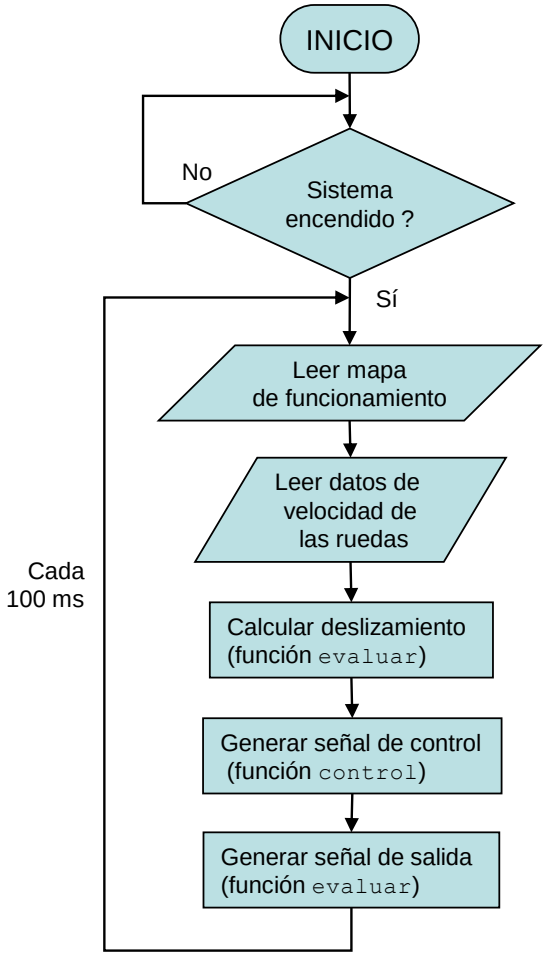


Figura 49: Diagrama de flujo general.

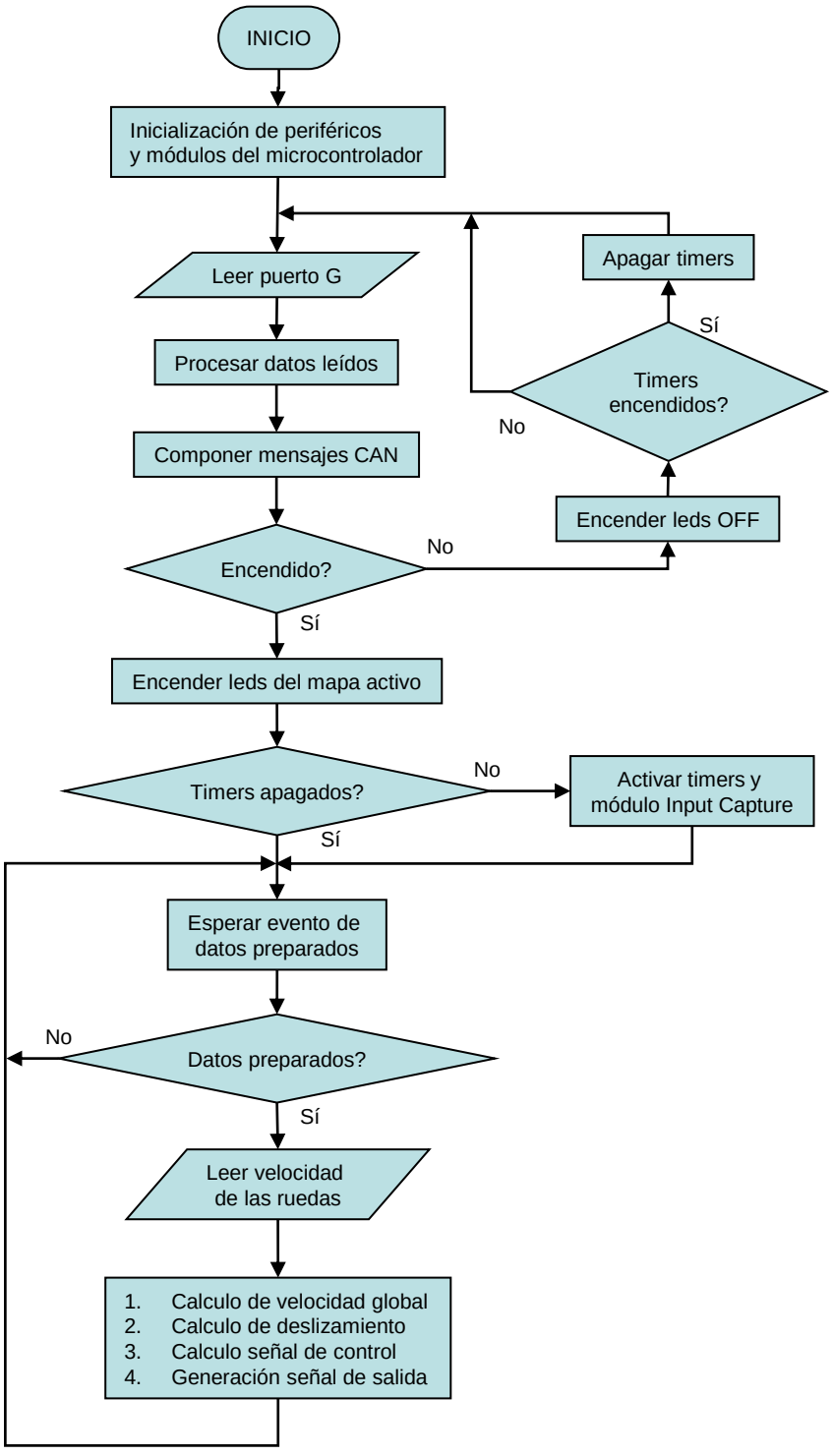


Figura 50: Diagrama de flujo de la función “main”.

Función del cálculo de deslizamiento

Cabecera:

- `float evaluar (float dd1, float di1, float td1, float ti1, float velocidad)`

Parámetros de entrada:

- `dd1`: velocidad rueda delantera derecha
- `di1`: velocidad rueda delantera izda
- `td1`: velocidad rueda trasera derecha
- `ti1`: velocidad rueda trasera izda
- `velocidad`: velocidad global actual del coche

Parámetros de salida:

- `error`: portentaje de deslizamiento calculado (expresado en tanto por uno)

Descripción:

- La función calcula el deslizamiento del vehículo en base a las velocidades de cada rueda y a la velocidad global del coche. Para su cálculo tiene en cuenta las siguientes variables globales:

```
unsigned char ALTAVELOCIDAD = 80; //Umbrales de velocidad en km/h.
unsigned char BAJAVELOCIDAD = 20;
unsigned char erroraltavelocidad = 10; //Errores permitidos. En %
unsigned char errormediavelocidad = 20;
unsigned char errorbajavelocidad = 80;
```

El diagrama de flujo de la función “evaluar” se muestra en la Figura 51.

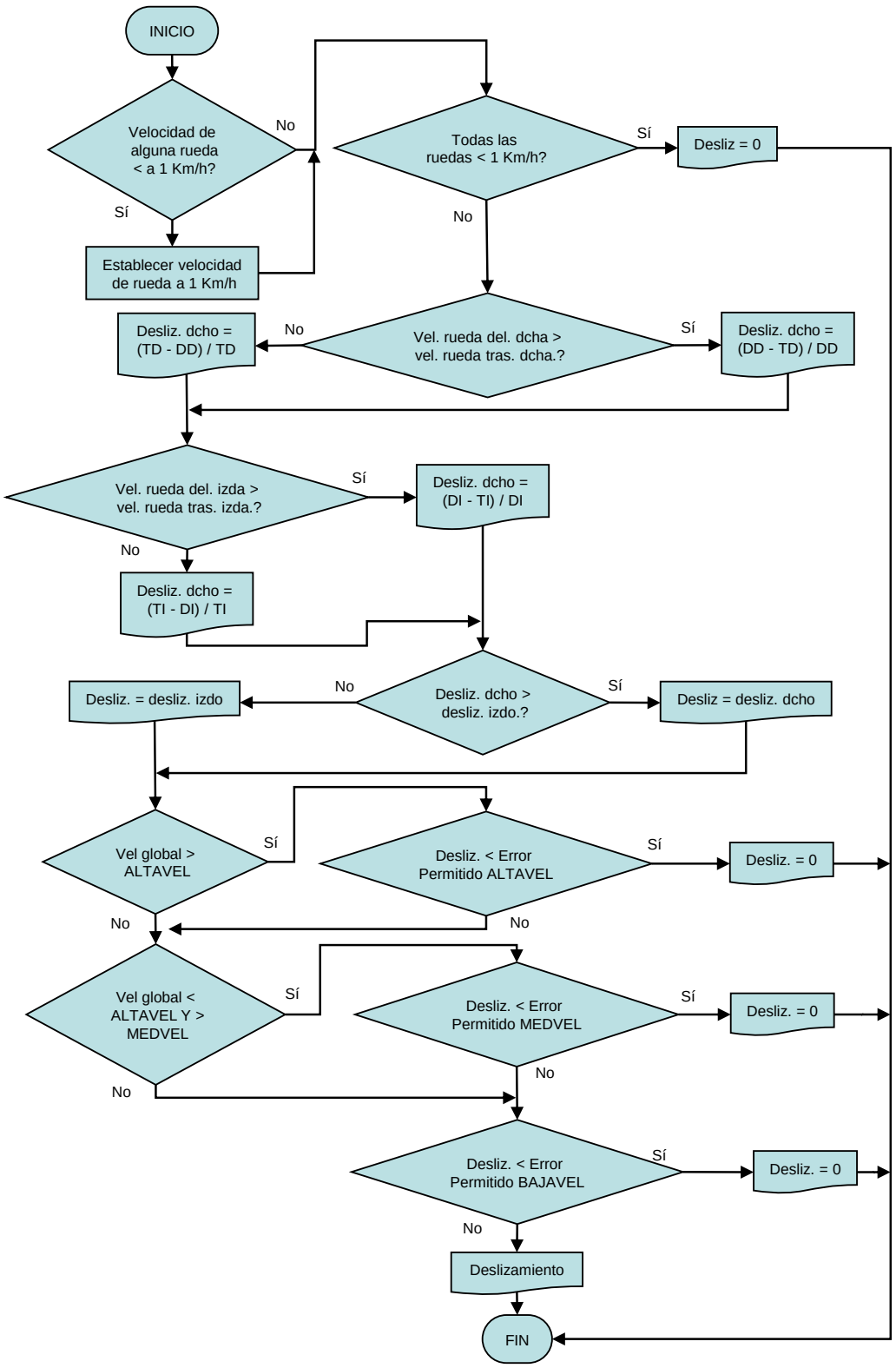


Figura 51: Diagrama de flujo de la función “evaluar”.

Función del controlador

Cabecera:

- `float control(float error,float Kp,float Ti, float velocidadglobal)`

Parámetros de entrada:

- `error`: error del sistema en tanto por uno.
- `Kp`: valor de la constante proporcional del regulador.
- `Ti`: valor del tiempo integral del regulador.
- `velocidadglobal`: velocidad global actual del coche.

Parámetros de salida:

- `salida`: valor de la señal de control.

Descripción:

- La función calcula el valor de la señal de control necesaria para la regulación del sistema. Para su cálculo tiene en cuenta las siguientes variables globales:

```
unsigned char BAJAVELOCIDAD = 20; //Umbrales de velocidad en
km/h.
unsigned char manteneraccion = 0; //Variable que sirve para
que la función del controlador sepa que tiene que seguir
aplicando señal de control aunque el error ya se haya hecho
cero.
```

El diagrama de flujo de la función “control” se muestra en la Figura 52.

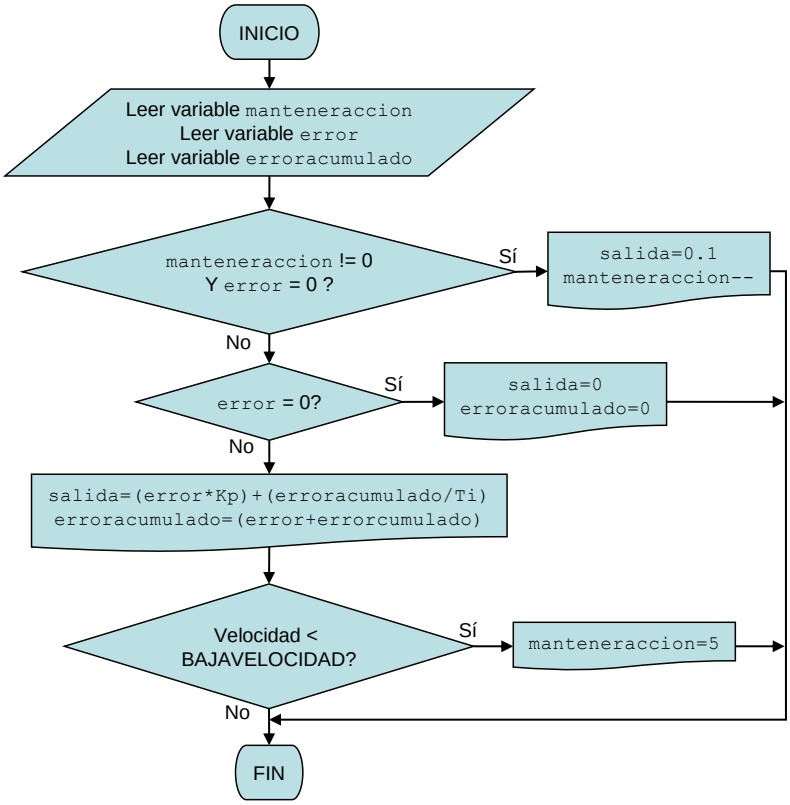


Figura 52: Diagrama de flujo de la función “control”.

Función del actuador

Cabecera:

- `void actuar(float salida)`

Parámetros de entrada:

- `salida`: valor de la señal de control.

Descripción:

- La función genera la señal de salida del sistema, la cual se enviará la centralita del vehículo.

El diagrama de flujo de la función "actuar" se muestra en la Figura 53.

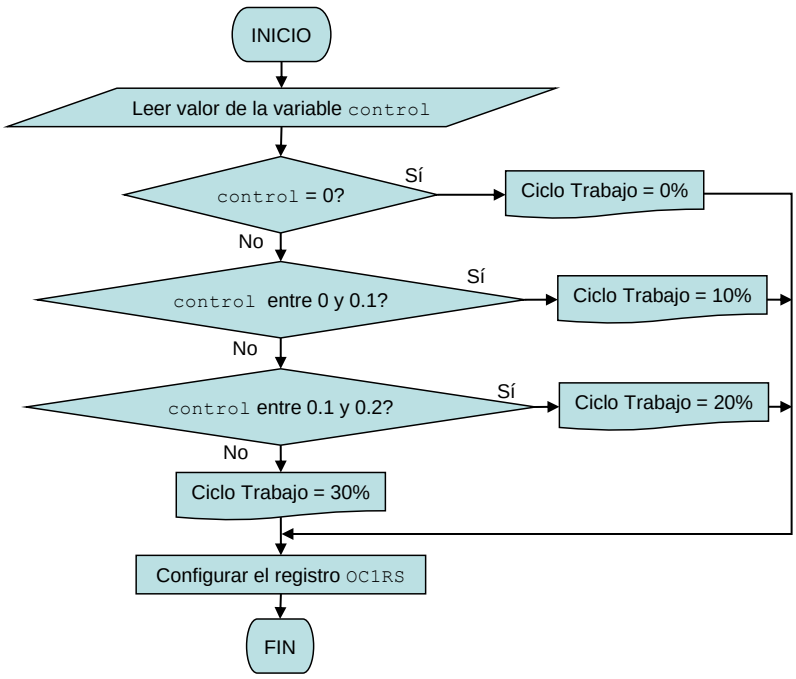


Figura 53: Diagrama de flujo de la función "actuar".

7.3. Pruebas de software

Durante el desarrollo del código del programa se han ido realizando pruebas para comprobar el correcto funcionamiento de éste.

La herramientas que esenciales que brinda el entorno de desarrollo MPLAB IDE para la realización de las pruebas del código han sido las siguientes:

1. Debugger MPLAP SIM. Es el depurador de software que trae MPLAB IDE. Con él es posible poner puntos de ruptura, ejecución paso a paso del código, medida del tiempo entre instrucciones, etc.

2. Ventana "Watch". Esta herramienta nos da la posibilidad de ver el valor que toman todas las variales declaras en el programa y los registros de funciones especiales (SFRs) del microcontrolador.

3. Estimulos. La herramienta "Stimulus" es esencial para simular señales de entrada en los pines del microcontrolador. Se ha usado, entre otras cosas, para la comprobación de la correcta medida del periodo de la señal aplicada al pin correspondiente del microcontrolador.

En el CD adjunto en el momento de la entrega de este libro, están incluidos todos los ficheros del proyecto creado en MPLAB IDE para el desarrollo del código. Entre ellos se encuentran las librerías del microcontrolador, el fichero fuente del código y el archivo de estímulos utilizado para las simulaciones.

8. Hardware: PCB e Interfaz con el piloto

8.1. Introducción al diseño y fabricación de la PCB

Para el diseño de la PCB he utilizado la suite de software ORCAD en su versión 10.5, por ser un software muy extendido y muy potente y además ya haber trabajado con él. Como contrapartida que tiene, es que es un software de pago y que requiere ciertos conocimientos para su manejo.

No voy a entrar en detalle en el uso de la suite ORCAD, más que nada porque sería extenderse mucho en la explicación de su manejo y porque en el mercado existen más soluciones para la creación de PCBs, algunas de ellas gratuitas y que pueden ser una buena opción a la hora de diseñar una PCB a nivel no profesional, como son: DesignSpark PCB y CadSoft Eagle PCB Design. Además, por si fuera poco, la suite de ORCAD ha ido evolucionando y actualmente, en su versión 16.6, su manejo difiere de la versión que yo voy a utilizar.

En cualquier caso, sí que haré un resumen extendido de cómo ha sido el diseño en ORCAD y de los pasos que consta y posteriormente explicaré y mostraré los esquemáticos del diseño para sostener su funcionamiento.

Diseño en ORCAD

El diseño, a groso modo, consta de dos etapas: por un lado, la generación del esquema de conexionado de los componentes electrónicos, lo que se conoce como "captura del esquemático" y por otro, el diseño de la PCB.

Para la captura del esquemático se utiliza el programa ORCAD CAPTURE, que se encuentra dentro de la suite ORCAD 10.5.

Para el diseño de la PCB se utiliza el programa ORCAD LAYOUT.

ORCAD CAPTURE:

- En Orcad Capture, generamos un nuevo proyecto, y en la hoja en blanco del proyecto vamos insertando componente a componente, creando las conexiones entre ellos mediantes “wires” y añadiendo los puntos de VCC y GND en los sitios correspondientes.
- En este mismo programa, debemos de darle a cada componente un nombre y un valor (Por ejemplo en el caso de una resistencia: nombre: R1, valor: 10k). También debemos asociar cada componente con un footprint.
- El footprint, en español: huella del componente, es una vista en planta de los pads, que servirán para soldar el componente a la PCB. La footprint también suele llevar inscripciones del nombre del componente, su valor y una línea que delimita su contorno, que sirve para generar un plano de montaje que ayude a la colocación del componente una vez creada la PCB.
- Aunque existen muchos footprints ya creados, es necesario algunas veces, crear tu propio footprint, esto se hace accediendo a la herramienta Library Manager, desde el programa Orcad Layout.
- Una vez que tenemos el conexionado de los componentes con su footprint asociada en Orcad Capture, hay tres pasos que se deben dar:
  - Hacer un chequeo de reglas eléctricas: DRC
  - Generar la lista de componentes: BOM
  - Y lo más importante, generar el Netlist. El netlist es un fichero que guarda información de todos los componentes y conexionado de nuestro circuito, así como la vinculación de la footprint a cada componente y que es necesario para que el programa Orcad Layout cargue esta información para poder comenzar a hacer la colocación de los componentes en la PCB. En este paso

es importante marcar una opción llamada “Run Eco to Layout” que sirve para que se comuniquen el programa ORCAD Capture y el Orcad Layout.

ORCAD LAYOUT:

- El diseño de una PCB con Orcad Layout es un diseño por capas. Diseñamos capas en las que vamos añadiendo los footprints, pistas, vías, etc.
- La PCB se compone de los footprints de los componentes (los cuales llevan los pads y otras inscripciones y dibujo de los bordes del componente) y de las pistas que unen los pads. En el caso de ser necesario también se insertan “Vías” para los cambios de capa de las pistas y “jumpers”.
- También es necesario para generar la PCB, tener un borde de placa, llamado “Board outline”, que se corresponde con el tamaño, o por lo menos con la delimitación en componentes y pistas, de la PCB que queremos generar.
- Una PCB, dependiendo del tipo de fabricación, puede tener componentes en la cara de arriba (llamada cara Top) – diseño a una cara - o en la cara de arriba y en la de abajo (llamada cara Bottom) – diseño a dos caras -.
- Las pistas que unen los pads de los componentes pueden ser colocadas, o como se suele denominar: ruteadas o trazadas, en la cara Top, en la Bottom, o en fabricaciones más complejas, en otras caras intermedias, que se encontrarían entre estas dos, haciendo una especie de sándwich con las diferentes capas, por visualizarlo de alguna manera.
- Por ejemplo, si diseñamos un circuito que solo lo fabricaremos en una cara, tanto pads como pistas deben de ir en esa cara. Si usamos componentes de inserción, los cuales necesitarán de taladros para poder pasar las patillas y luego soldar el pad, se usaría la cara bottom, es decir, la cara de abajo, para los pads y las pistas, y los componentes irían en la cara de arriba.

- Si los componentes fueran de montaje superficial (SMD), usaríamos la cara Top para pads y pistas.
- Orcad Layout tiene multitud de capas siendo las más importantes las siguientes:
  - Capa TOP: Es la capa que lleva el material de cobre, lo que serían pads o pistas de cobre que se quieran poner en la cara de arriba. No confundir cara con capa, en la cara TOP, o cara de arriba, pueden generarse multitud de capas como veremos a continuación, aunque la mayoría de veces se usa la expresión "Cara top", para referirnos a la capa TOP.
  - Capa BOT: Idéntico a lo anteriormente descrito pero para la cara de abajo o cara bottom.
  - Capa ASYTOP y ASYBOT: Capas para la serigrafía. Tanto en la cara de arriba como en la de abajo, podemos generar capas de serigrafía, en las que se imprimen textos y dibujos. Las capas ASYTOP (que va en la cara de arriba), como la ASYBOT (en la cara de abajo) suelen llevar serigrafiadas el nombre y valor de los componentes.
  - Capa SSTOP y SSBOT: Son también capas de serigrafía. Estas suelen tener dibujado el contorno de los componentes, se le conoce también como plano de montaje porque ayuda a colocar los componentes.
  - DRILL: Capa con la información para hacer los taladros en la PCB
  - GLOBAL LAYER: Esta capa se usa para contener el borde de placa.
  - INNER1,2,3...: Las capas inner son capas que se encuentran entre las capas de arriba y de abajo, en diseños complejos, sirven para llevar pistas y que sea más fácil el ruteado. Se consiguen con procesos de fabricación profesionales.

En diseños no profesionales, solo trabajamos con la capa que lleva el cobre (ya sea top o bottom), pero en la fabricación profesional tenemos capas intermedias y capas de serigrafía que sirven para imprimir en la PCB los nombres y valores de los componentes así como su contorno. Cada trabajo de serigrafía, requiere de una capa para su posterior fabricación.

En nuestro caso, haremos uso de la cara de arriba y de abajo – diseño a dos caras – y pondremos tanto pistas como pads en ambas caras, por lo que haremos uso de las capas TOP y BOT.

Debido al número de componentes y para facilitar en gran medida la colocación de ellos y el trazado de las pistas, he decidido colocar los componentes SMD en la cara de arriba y los de inserción en la de abajo, esto hará que la mayoría de pads estén solo en la capa TOP así como las pistas y se disminuirán en importante medida las vías entre capas.

Esto requiere de un importante detalle: Los footprints de los componentes de inserción, están diseñados para su colocación en la cara top, dejando los pads en la cara bottom. Al querer que los pads estén en la cara top, para unirlos fácilmente mediante pistas con los pads de los componentes SMD, tenemos que aplicar la opción OPOSITE a los footprints de los componentes de inserción. Lo que hace esta opción del programa Orcad Layout es invertir el footprint de tal forma que los pads queden en la cara top y que no se desordenen los números de los pines.

En cuanto al ruteado de las pistas, Orcad Layout tiene una opción de autoruteado, en la que una vez que tenemos colocados todos los footprints de los componentes, él genera un proceso para trazar las pistas entre los pads de los componentes siguiendo una serie de reglas preconfiguradas que vienen a ser: ancho de las pistas, distancia mínima entre pistas y distancia mínima entre pistas y pads. El proceso de autoruteado es un proceso en el que para obtener resultados satisfactorios debemos de vigilar muchos parámetros, es recomendable, primero, trazar de forma manual las pistas más importantes, que son las que llevan la alimentación, para minimizar su longitud y optimizar el recorrido y luego comenzar con el proceso de autoruteado. En cualquier caso, para obtener unos mejores resultados y siempre que se pueda, es conveniente estudiar bien la colocación de los componentes y trazado de las pistas y hacer un trazado a mano de ellas. En el caso de la PCB que he fabricado, la diferencia entre los

diferentes trazados automáticos llevados a cabo y el trazado manual que finalmente se ha hecho, era enorme.

8.2. Partes del diseño y esquemáticos

Con todos los datos que ya se han dado sobre el sistema, ya solo queda diseñar los circuitos electrónicos que harán que todo funcione según lo estudiado.

Antes de nada, vamos a hacer una lista de las conexiones mínimas que requiere el microcontrolador así como otras indicaciones para su puesta en funcionamiento, que podemos sacar del manual del mismo:

- Conexiones mínimas:
  - Debemos conectar todas los pines de masa (Vss) y tensión positiva VDD
  - Los pines AVDD y AVSS
  - Se debe conectar el pin de Reset MCLR y elaborar un circuito de reset.
  - Conectar los pines PGEC1 y PGED1 que son usados para la programación del microcontrolador.
- Indicaciones de uso:
  - El microcontrolador se alimenta a +3,3V
  - Entre cada par de pines de Vss y Vdd se debe conectar un condensador de desacoplo de un valor recomendado de 100nF y con un bajo valor de ESR. Se recomienda un condensador cerámico.
  - Entre cada par de pines de AVss y AVdd se debe conectar un condensador de desacoplo de un valor recomendado de 100nF y con un bajo valor de ESR. Se recomienda un condensador cerámico.

- Los condensadores de desacoplo deben situarse lo más próximos al microcontrolador que se pueda.
- Es necesario colocar un condensador de desacoplo para el pin VCAP/VDDCORE. Es te condensador debe ser entre 4.7uF y 10uF de 16V de tensión mínima y un ESR por debajo de los 5 ohmios. Este condensador es recomendado que se coloque a menos de 6 mm del pin VCAP/VDDCORE.

Centrándonos en el diagrama de bloques presentado en el apartado 6.9, iremos presentando los diferentes circuitos que hemos diseñado para que realicen la tarea correspondiente.

8.2.1. Alimentación

La tensión de alimentación del dispositivo es de +12V, pero tendremos que trabajar internamente con los +3,3V del microcontrolador y los +5V que requieren otros sistemas para su correcto funcionamiento.

Para ello hemos diseñado un circuito de regulación de tensión en dos etapas, primero convertiremos la señal de +12V a +5V a través de un regulador conmutado y después de +5V a +3,3V por medio de un regulador lineal de tensión.

Podemos ver el esquemático en la Figura 54.

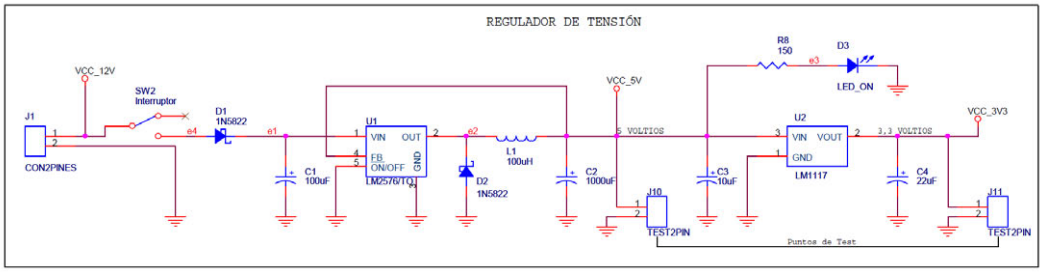


Figura 54: Circuito de alimentación.

8.2.2. Microcontrolador, programación y reset

En la Figura 55 podemos ver dibujado el micocontrolador con todos sus pines y conexiones, así como el circuito de reset y el de programación

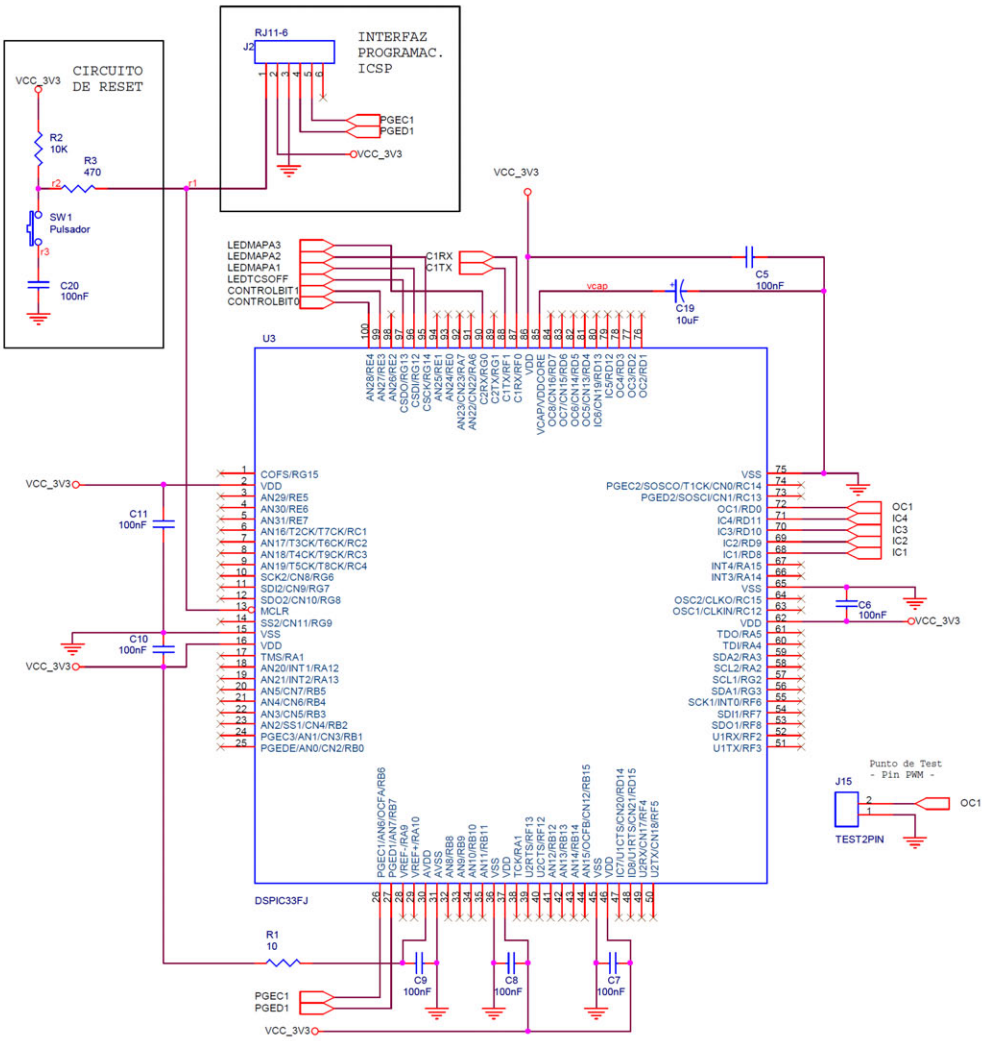


Figura 55: Esquemático del microcontrolador.

8.2.3. Bus CAN

La parte del bus CAN requiere de un transceiver y un conector DB9. Podemos ver el circuito en la Figura 56.

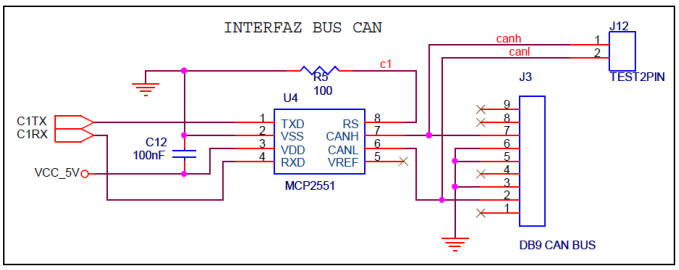


Figura 56: Circuito bus CAN.

8.2.4. Circuito de control remoto de la PCB principal

Lo podemos ver en la Figura 57

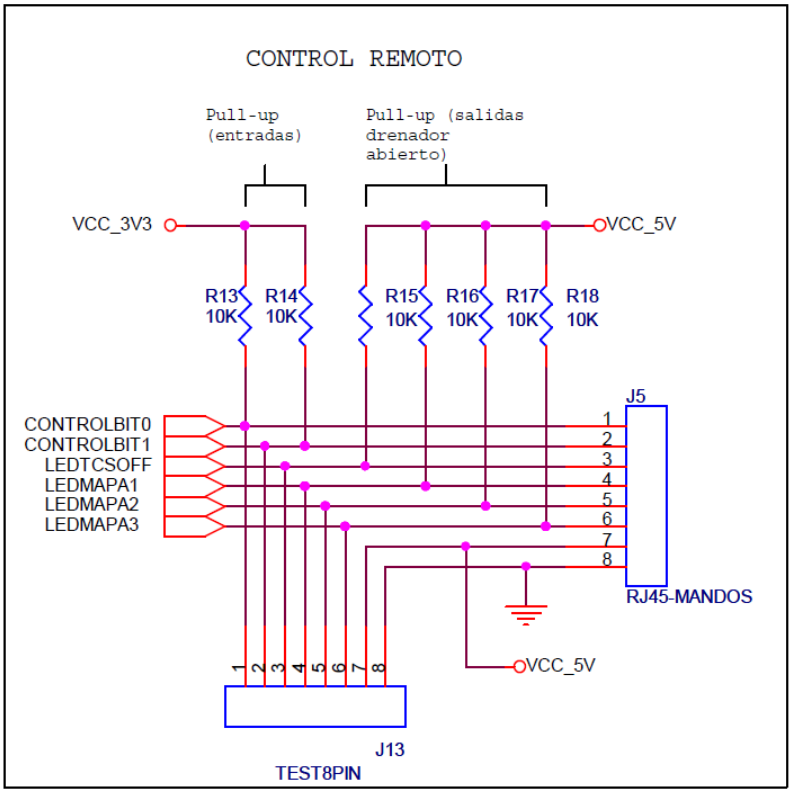


Figura 57: Circuito de la parte del control remoto en la PCB principal.

8.2.5. PCB del control remoto

El control remoto requirió de la fabricación y diseño de otra PCB. En la Figura 58 podemos ver su esquemático.

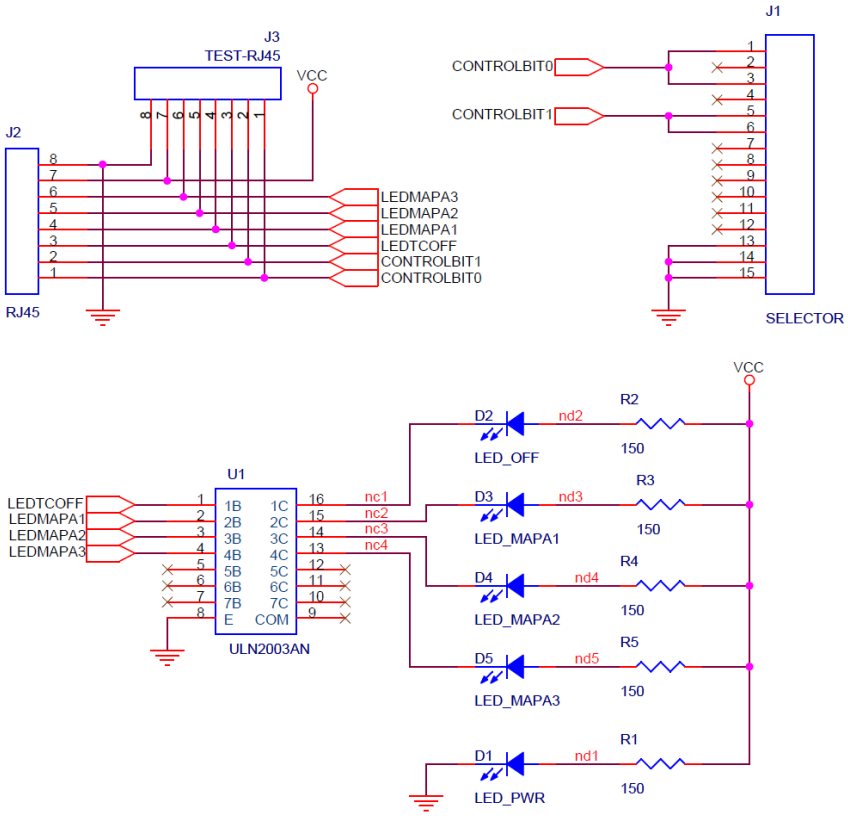


Figura 58: Circuito del mando de control remoto.

8.2.6. Circuito salida PWM

Para el diseño de este circuito hemos tenido en cuenta que la salida del microcontrolador es de 3,3 voltios y la salida del sistema tenía que llegar a 5 voltios. Para conseguir subir el nivel de tensión hemos incluido un circuito con un comparador de tensión. En la Figura 59 podemos ver el circuito.

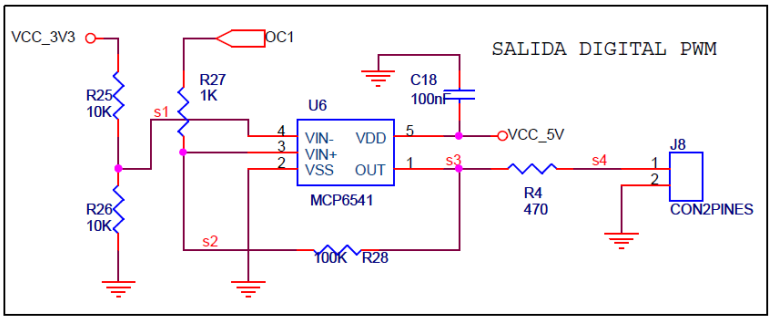


Figura 59: Circuito de adecuación de la señal PWM.

8.3. Proceso de fabricación

El proceso de fabricación que vamos a utilizar para fabricar la PCB será mediante insolado.

Al fabricar la PCB bajo el método del insolado, nuestro objetivo es llegar a generar un fotolito para su uso con la insoladora.

Una insoladora es una máquina con forma de caja y que en su interior se genera luz. Una placa de cobre barnizada con un material fotosensible se mete dentro de la insoladora cubierta con el fotolito para exponerla a la luz durante un tiempo concreto.

Un fotolito viene a ser una hoja transparente con nuestro diseño impreso. Al sobreponerlo en la placa de cobre fotosensible y meterlo en la insoladora para que le dé luz durante un tiempo determinado, las zonas que están debajo de lo impreso (pads y pistas) no se verían afectadas por la luz, lo que hace que con un proceso químico posterior, podamos quedarnos solo con el cobre que no se ha visto expuesto a la luz.

Cuando sacamos la placa de cobre de la insoladora, primero debemos eliminar el barniz fotosensible en un proceso llamado revelado. Posteriormente en un proceso

químico llamado atacado, se elimina el cobre que no está protegido con el barniz, quedándonos así con los pads y las pistas del diseño.

8.3.1. Fotolitos

El fotolito lo generamos con el programa ORCAD LAYOUT. Mediante el programa ORCAD LAYOUT podemos imprimir cada una de las capas que hemos diseñado para generar el fotolito. En la Figura 60 podemos ver el fotolito de la capa top de la PCB.

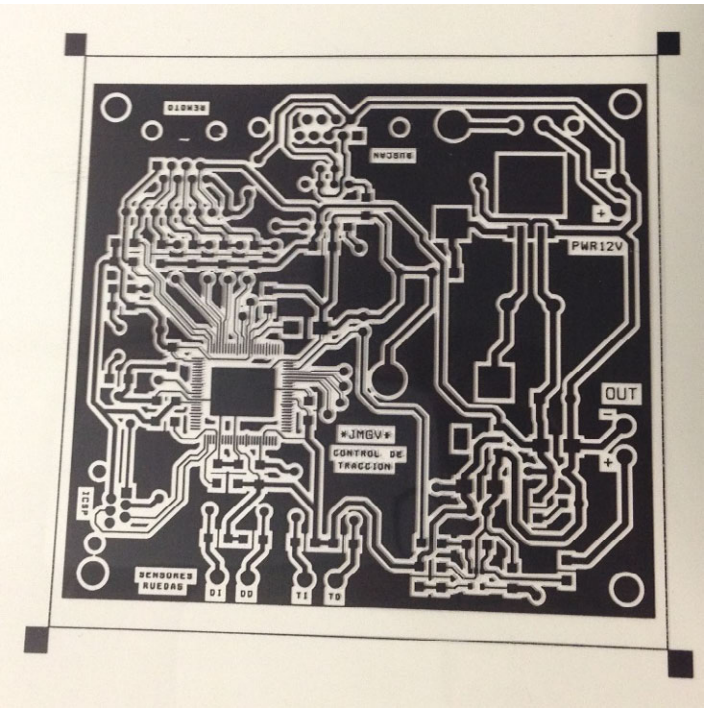


Figura 60: Fotolito de la capa top.

Nosotros imprimiremos la capa top y la capa bottom puesto que nuestro diseño es un diseño a dos caras. Para generar los fotolitos y para su posterior uso, es bueno seguir estas recomendaciones:

1. Utilizar papel de acetato transparente apto para impresoras láser. A estos papeles se adhiere muy bien el polvo del tóner y no tendremos problemas con borrado de

pistas. Además, si se conservan bien, podemos utilizarlos más de una vez para generar más PCBs.

2. Imprimir más de un fotolito para usarlo uno encima de otro. Con esto conseguiremos más opacidad en pistas y pads lo que a la hora del insolado es muy recomendable.

3. Cuando diseñamos las capas es imprescindible insertar marcas para alinearlas, sobre todo en nuestro diseño ya que tendremos que alinear los fotolitos de la capa top y de la bottom y poner en medio de ellos la placa de cobre. Estas marcas también deben servir para saber el lado correcto de colocación, ya que imprimiremos en una hoja transparente y podemos perder la orientación. Estas marcas se conocen como marcas fiduciales.

4. Cuando colocamos el fotolito encima de la placa de cobre, tenemos que cerciorarnos que la cara del fotolito donde la impresora ha depositado el polvo del toner es la cara que va a quedar pegada al cobre, de esta manera, minimizamos al máximo los problemas de que la luz pueda entrar por los contornos de las pistas y pads. Para ello se invierte la imagen del fotolito de la capa top antes de imprimirlo. La insoladora que hemos utilizado, además dispone de bomba de vacío, lo que hace que el fotolito se pegue a la placa de cobre y no deje ningún espacio para que la luz penetre por los laterales de las pistas y pads.

8.3.2. Insolado

Depende del espectro y potencia de la luz que genere la insoladora y de la calidad del material fotosensible que recubra la placa de cobre, los tiempos de exposición varían.

En nuestro caso con 90 segundos ha sido suficiente.

La insoladora utilizada (Figura 61 disponía de puntos de luz en la parte superior e inferior, lo que posibilita la creación de una PCB de doble cara en un solo paso.

Antes del insolado es conveniente limpiar bien las placas transparentes donde se apoya la PCB y la plancha que la recubre para que no haya ninguna suciedad. El otro paso crítico es alinear perfectamente los fotolitos.

Figura 61: Insoladora.

8.3.3. Revelado y atacado químico

Una vez que la placa de cobre está insolada, hay que llevar a cabo dos pasos:

1. Revelado: El revelado, elimina el material fotosensible que ha sido expuesto a la luz en el proceso de insolado. Se hace sumergiendo la placa de cobre en una mezcla de sosa caustica y agua.

   En nuestro caso, y con la concentración de sosa caustica en el líquido revelador que nos han brindado en el laboratorio de la universidad, han sido suficientes 40 segundos para completar el revelado. Un ligero movimiento es recomendable para eliminar el material fotosensible.

   Después del revelado es conveniente sumergir la placa en agua para limpiar los restos de sosa caustica antes de llevar a cabo el siguiente paso.

2. Atacado químico: El atacado químico, elimina el cobre que no está protegido por el material fotosensible y que eliminamos en el paso anterior.

   Para el atacado químico, sumergimos la placa en una mezcla de ácido clorhídrico, agua y peróxido de hidrógeno (agua oxigenada). La mezcla del laboratorio estaba compuesta de 50 % de ácido clorhídrico, un 37,5 % de agua y un 12,5 % de agua oxigenada (otra forma de decirlo es: 1 parte de agua oxigenada, 3 partes de agua y 4 partes de ácido clorhídrico).

   En este paso es conveniente agitar la placa de cobre para acelerar el proceso. En nuestro caso, en torno a 4:30-5 minutos fueron necesarios para eliminar el cobre.

   Después del paso del atacado químico debemos aclarar la placa con agua, secarla y eliminar el barniz fotosensible que queda porque si no, no podremos medir continuidad ni soldar en los pads. El material fotosensible se elimina perfectamente con acetona.

Figura 62: Revelado y atacado químico.

8.3.4. Pruebas visuales y de continuidad

Después del revelado, atacado y limpieza de la placa, pasamos a su inspección visual, es importante tener un esquema del diseño al lado y poner atención en encontrar cortocircuitos o circuitos abiertos.

Para comprobar a fondo la continuidad de las pistas del circuito hacemos uso de un polímetro. La función de continuidad de un polímetro con la capacidad de emitir un pitido cuando existe continuidad nos vendrá fenomenal es este paso.

Si nos encontramos algún cortocircuito podemos hacer uso del cuter o alguna herramienta tipo dremel para solucionarlo. En cambio si lo que nos ha pasado es que tenemos algún circuito abierto, tendremos que solucionarlo aplicando estaño e incluso añadiendo algún hilo de cobre.

Problemas encontrados: En nuestro caso, la primera PCB la tuve que desechar porque los pads del microcontrolador eran demasiado grandes y había muchos cortocircuitos que no pude solucionar con el cuter. Se podría decir que el footprint del microcontrolador era demasiado generoso a tenor de la calidad de nuestro proceso de fabricación (sobre todo porque al usar dos fotolitos, uno encima de otro, las pistas y pads ensanchaban mínimamente). Para solucionarlo, reducí los pads del footprint y la anchura de las pistas más cercanas al microcontrolador, así como cambié de acetato e impresora para la realización de los fotolitos. La segunda placa quedó espectacular, tan solo hicieron falta dos puntos de estaño en dos pistas que había perdido continuidad. En la Figura 63 se puede ver el resultado de las pistas y pads del microcontrolador.

8.3.5. Taladros

El siguiente paso es la realización de los taladros.

Los componentes de inserción y los puntos de fijación de la PCB requieren de taladros. Usamos un taladro de banco disponible en los laboratorios de la universidad (Figura 64) con brocas de distintos diámetros. Es un paso sencillo que requiere de paciencia para centrar bien los taladros en los pads y no romper las brocas más finas con las que trabajemos, que pueden ser de 0.6 mm de diámetro o incluso menos.

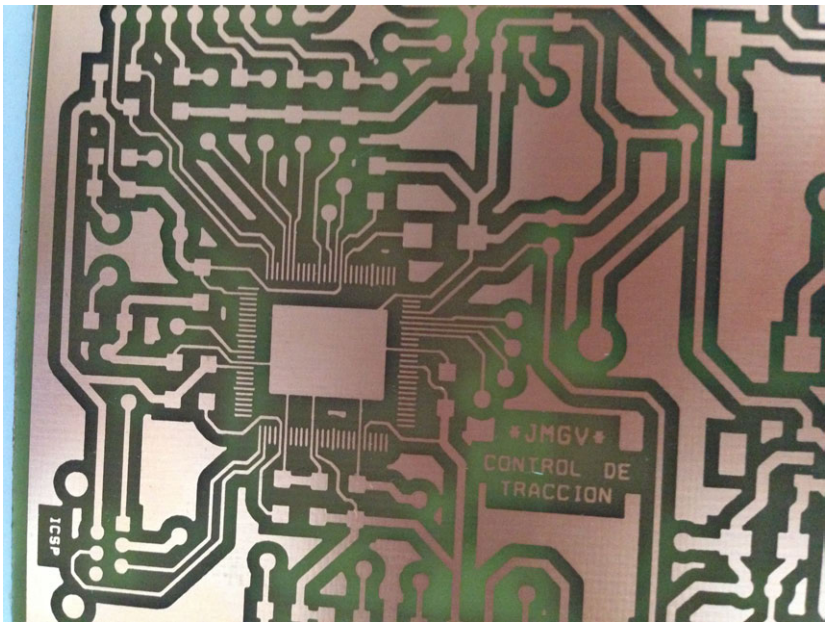


Figura 63: Pistas y pads en la zona del microcontrolador.

8.3.6. Montaje y soldadura de componentes

Una vez efectuados los taladros pasamos al paso de la soldadura de los componentes, es importante soldar primero los componentes SMD y después los de inserción, al ser los primeros, más difíciles de soldar.

Soldadura de componentes SMD

Los componentes SMD o de montaje superficial, van colocados encima de los pads, en la misma cara que éstos. No requieren de taladros, simplemente se sueldan los pines a los pads con la ayuda del estaño.

El componente más difícil de soldar ha sido el microcontrolador, todo un reto: un encapsulado TQFP de 100 pines en un cuadrado de 14mm de lado. Si no se ha realizado nunca, primero es conveniente soldar algunas resistencias y condensadores SMD para ir cogiendo el punto del tiempo de aplicación de calor con el soldador y la cantidad de estaño.

Figura 64: Taladro de banco.

Para la soldadura, se utilizó un soldador de punta fina, hilo de estaño muy fino (0,4mm de diámetro), una bomba de vacío manual (para absorber el estaño cuando nos pasamos de cantidad) y flux, el maravilloso flux... El flux es un material que disminuye el punto de fusión del estaño y el cobre y ayuda a que la soldadura sea más rápida, más limpia y más fácil. Literalmente, al aplicar flux sobre la soldadura antes de proceder a ella, el estaño parace que "viaja" hasta el pad para depositarse allí y hacer que el pin y el pad queden unidos.

Algunos consejos para soldar componenets SMD complejos son los siguientes:

1. Aplicar flux en los pads y pines antes de soldar. El flux que hemos usado es un flux líquido de la marca JBC que viene con un aplicador de pincel.
2. Colocar el componente perfectamente alineado con los pads y soldar solo un par de esquinas para que no se mueva.
3. Por si no lo he dicho... aplicar bien de flux!
4. Poner el hilo fino de estaño encima de los pines de un lado del integrado, cubriéndolos todos y pasar lentamente, pero sin pausa, la punta del soldador

tocando los pines uno a uno, calentándolos y haciendo así que el estaño se deposite en el pad y una los pines a ellos.

Es un proceso tedioso, pero con paciencia se puede conseguir muy buen resultado, como se puede ver en la foto de la Figura 65

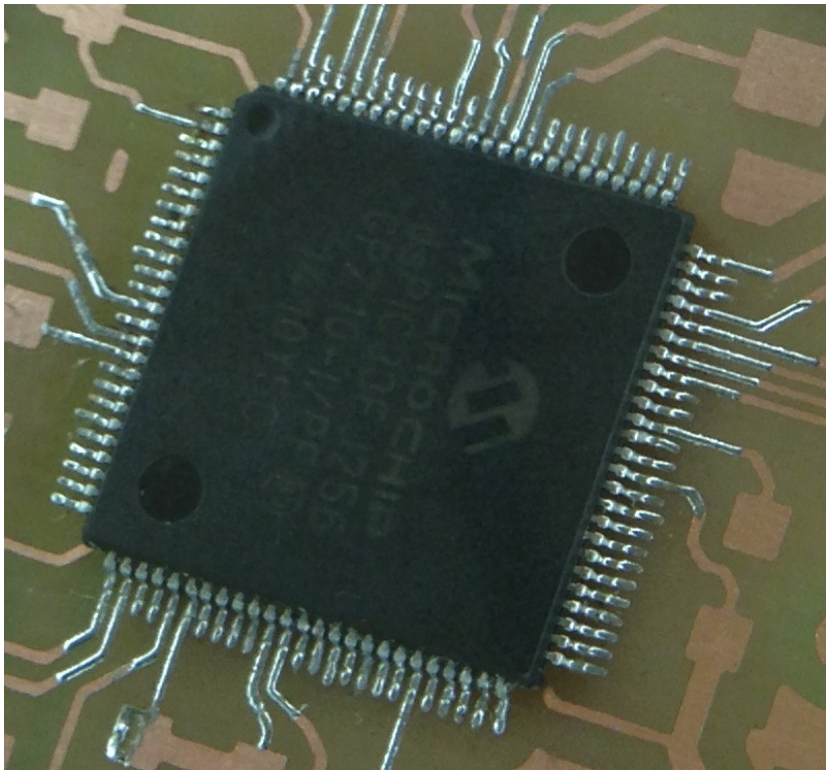


Figura 65: Soldadura del microcontrolador.

Componentes de inserción y vías

Los componentes de inserción, habiendo pasado por la parte anterior, se sueldan fácilmente. Tenemos que insertar los componentes firmemente por la cara contraria a los pads, e ir soldando patilla a patilla.

Las vías, que son cambios de cara de las pistas, se realizan pasando un hilo de cobre de un diámetro que haga que quede firme en el taladro que hemos realizado en el pad y soldándolo igual que si se tratara de una patilla de un componente de inserción, pero por ambas caras.

En las fotos de las Figuras 66 y 67 se puede apreciar la PCB terminada.

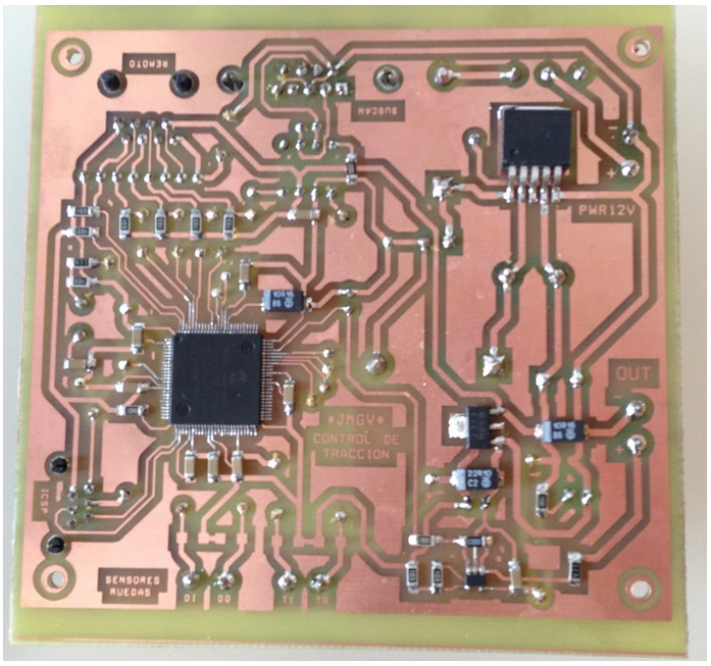


Figura 66: PCB cara top.

Tambien podemos ver en la Figura 68 la PCB del mando de control remoto acabada.

8.3.7. Ficheros de fabricación

Los fotolitos y los ficheros de fabricación para la realización de la PCB se encuentran en el CD entregado junto con los libros de este PFC.

Así mismo se podrá encontrar el listado de los componentes con su código RS (empresa especializada en la venta por internet de componentes electrónicos).

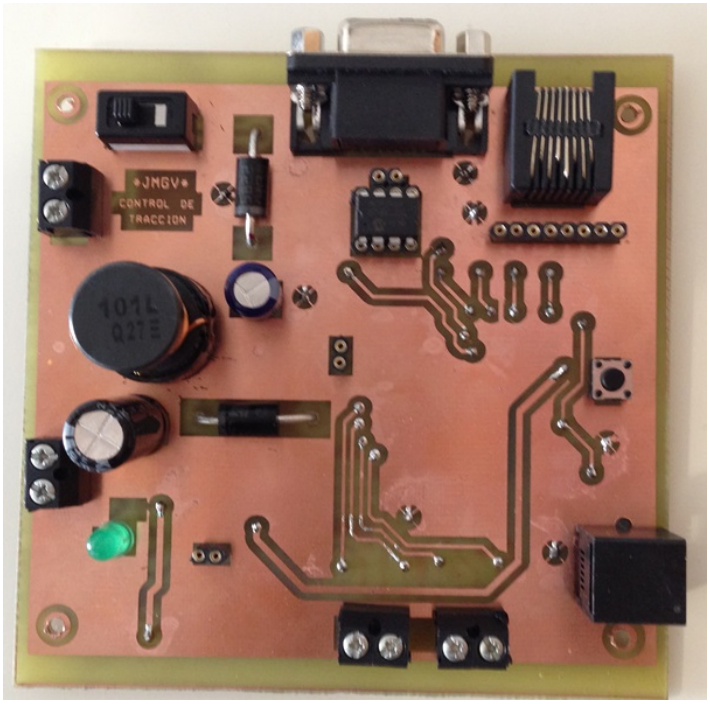


Figura 67: PCB cara bottom.

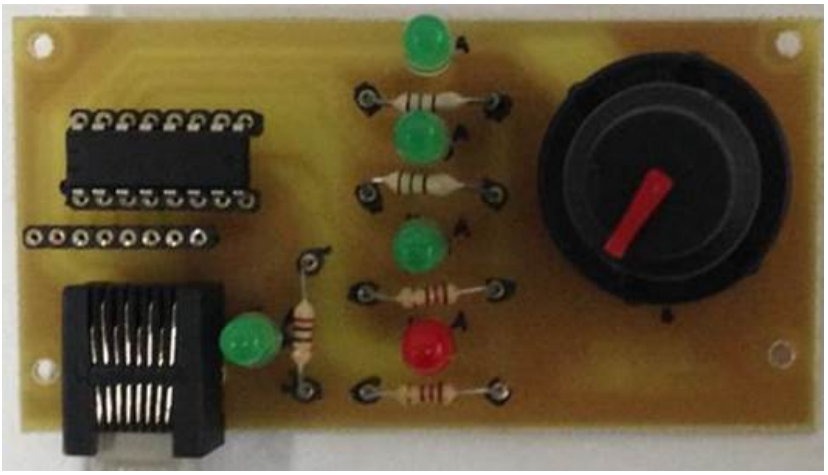


Figura 68: PCB del mando de control remoto.

9. Pruebas

9.1. Kit de depuración de Microchip

Para programar el microcontrolador de la PCB usaremos un dispositivo de programación de Microchip llamado MPLAB ICD2. Este dispositivo se conecta al ordenador mediante puerto USB y a nuestra PCB mediante conector RJ11. Una vez conectado al ordenador, desde la configuración del entorno de desarrollo MPLAB IDE, lo configuramos correctamente y procedemos a la programación del microcontrolador. Los pasos a realizar son los siguientes:

1. Abrir MPLAB IDE
2. Conectar MPLAB ICD2 al ordenador mediante el cable USB
3. Se abre un asistente de configuración en el entorno MPLAB IDE. Básicamente nos ayuda a configurar el dispositivo MPLAB ICD2, seleccionando la interfaz de conexión (en este caso USB) y habilitando la descarga del software para su correcta configuración.
4. Conectamos mediante el cable RJ11 suministrado el MPLAB ICD2 y nuestra PCB y encendemos la PCB.
5. En MPLAB IDE elegimos el dispositivo MPLAB ICD2 como programador: Menú "Programmer"-"Select programmer"-"MPLAB ICD2"
6. Por último, pulsamos el botón de programar o elegimos en el menú "Programmer" la opción "Program".

En cuanto a la parte de debugger, para las pruebas del programa, el dispositivo MPLAB ICD2 nos permite hacer uso de la ejecución de instrucciones paso a paso, poner puntos de ruptura y leer el valor de las variables en tiempo de ejecución. Para seleccionar

el dispositivo MPLAB ICD2 como debugger, debemos ir al menú "Debugger"-"Select Tool" y pinchar en "MPLAB ICD2". Los demás pasos necesarios son análogos a la parte de programación. Para iniciar el debugger debemos pinchar sobre el botón Run ("Debugger"-"Run"). Una de las opciones esenciales cuando se utiliza el debugger es ver el valor que toman las variables del programa y los valores de los registros del microcontrolador, esto lo hacemos con la herramienta "Watch": Menú "View"-"Watch".

9.2. Voltajes de alimentación

La PCB se alimentará de la batería del vehículo, que tiene un voltaje típico de +12V, pero es posible que por la acción del alternador del vehículo esta tensión sea mayor llegando a los +14V, en cualquier caso, el regulador conmutado utilizado como primera etapa de la regulación de tensión está preparado para trabajar con tensiones de entrada incluso mayores.

En las pruebas realizadas a la parte de la alimentación de la PCB, se ha comprobado que los voltajes generados por los dos reguladores de tensión utilizados en la PCB, han sido de +5V y +3,3V tal y como se esperaba.

9.3. Pruebas del control remoto

Tras comprobar visualmente y con el polímetro que el resultado de la fabricación de la PCB del control remoto era satisfactorio, pasamos a comprobar su funcionamiento por separado.

Para ver el esquemático de la PCB ir al apartado 8.2.5.

Primero comprobamos exhaustivamente con el polímetro que no existe ningún cortocircuito y después, haciendo uso de los 8 pines de test situados al lado del conector RJ45, medimos la tensión entre los pines 1 y 2 (Controlbit0 y Controlbit1 respectivamente) y masa (pin 8). Hacemos la misma medición para cada una de las posicione del selector y obtenemos las mediciones que muestra la Tabla 17.

	Pin 1 (Controlbit0)	Pin 2 (Controlbit1)
Posición 1	Cortocircuitado con pin 8 (GND)	Cortocircuitado con pin 8 (GND)
Posición 2	Circuito abierto	Cortocircuitado con pin 8 (GND)
Posición 3	Cortocircuitado con pin 8 (GND)	Circuito abierto
Posición 4	Circuito abierto	Circuito abierto

Tabla 17: Mediciones de la PCB del mando remoto.

El funcionamiento es el correcto y con ello, haciendo uso de dos pines de entrada del microcontrolador (RE4-pin100- y RE3-pin99-) y unas resistencias de pull-up, podremos diferenciar 4 estados lógicos:

- Posición 1: TCS apagado
- Posición 2: TCS activo MAPA1
- Posición 3: TCS activo MAPA2
- Posición 4: TCS activo MAPA3

Otra prueba que haremos, es simular que se encienden los leds y así comprobamos tanto el funcionamiento de éstos como el del integrado ULN2003AN (array de transistores Darlington) que suministra la corriente necesaria para su activación. Para ello, con ayuda de una protoboard y un cable especial que hemos fabricado (cable de 8 hilos que por un lado tiene conector RJ45 macho y por otro lado pines de test) seguimos estos pasos:

1. Conectamos el conector RJ45 a la PCB y los pines de test a la protoboard

2. Generamos 5V y lo aplicamos a uno de los pines de una resistencia de 10K, haciendo las veces de resistencia de pull-up, que es como se enviará la señal desde la PCB principal.

3. El otro extremo de la resistencia lo aplicamos a los pines 3, 4, 5 y 6 consecutivamente y apreciamos como se van encendiendo los leds.

NOTA: Uno de los led, el ledpower, que está al lado del conector RJ45, permanece encendido mientras la PCB de control remoto esté alimentada. Este led no pasa por el integrado ULN2003AN y en las pruebas comprobamos cómo se encendía correctamente.

En la imagen de la Figura 69 vemos como con ayuda de la protoboard y el cable RJ45 de test, logramos encender uno de los leds y el led de power.

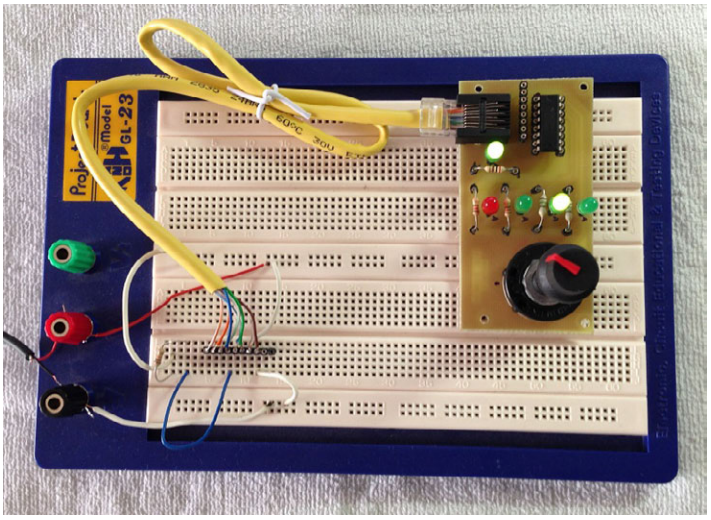


Figura 69: Pruebas del mando de control remoto.

Problema encontrado: Uno de los leds no se encendía y comprobando bien las soldaduras me di cuenta que no hacía buen contacto. La resoldadura solucionó el problema.

La última prueba es conectar el control remoto a la PCB principal y encenderla. Según el funcionamiento del sistema, con el selector en posición 1, el ledtcsoff se debe encender y los demás permanecer apagados (salvo el ledpower). El posición 2

se enciende el ledmapa1, en posición 3 el ledmapa2 y en posición 4 el ledmapa3. La comprobación se realizó con éxito.

Problema encontrado: En una primera prueba, los leds funcionaban justo al contrario de cómo deberían: el led que señalaba el selector permanecía apagado y los demás permanecían encendidos. El problema era que los pines de salida del microcontrolador que gobiernan los leds estando configurados en drenador abierto funcionan de tal manera que con un "1" lógico significa que el pin está en alta impedancia (por lo que a través de la resistencia de pull-up llega corriente al control remoto para encender el led) y con un "0" lógico, el pin está a masa. Yo había confundido los valores y los trataba justo al revés. Tras un cambio de configuración en el programa, se solucionó este error.

9.4. Lectura de la velocidad de las ruedas

Haremos uso del debugger del dispositivo MPLAB ICD2.

Lo que vamos a hacer es simular una señal igual a la de los sensores de efecto Hall acoplados a las ruedas y comprobar que el sistema de control de tracción es capaz de leer correctamente esta señal y calcular la velocidad a la que va la rueda.

En la parte del programa, mostramos en el debugger la ventana "watch" para poder ver el valor de los registros y variables en tiempo de ejecución.

En la parte hardware, tenemos que hacer lo siguiente:

1. Generar una señal cuadrada de valor mínimo 0 voltios y máximo 3,3 voltios, de una frecuencia en torno a 390 hertzios.

2. Como en el entorno real de trabajo, la salida de los sensores de efecto hall es de colector abierto, en la PCB usamos una resistencia de pull-up para generar el voltaje de 3,3V al activarse la señal y posteriormente conectarla al pin de entrada del microcontrolador. Esta resistencia la tenemos que desoldar para inyectar directamente la señal proveniente del generador de señales que ya habremos ajustado previamente.

3. Haciendo referencia de nuevo al sensor de efecto Hall de las ruedas, de ellos, solo nos viene un cable, ya que se supone que la masa es común en todos los dispositivos del coche. Por esto, debemos conectar la masa del generador de señales a la masa de la PCB y el positivo del generador de señales a la entrada del sensor correspondiente en la PCB. Para la prueba usaremos alternativamente cada una de las entradas, para verificar que todas funcionan.

En las Figuras 70 y 71 podemos ver la señal que inyectamos a la PCB para simular los sensores de las ruedas.

Figura 70: Generador de señal.

En la Figura 72 podemos ver una captura de pantalla de la ventana del "watch"del debugger y la velocidad que el sistema calcula.

Calculo teórico de la velocidad de la rueda:

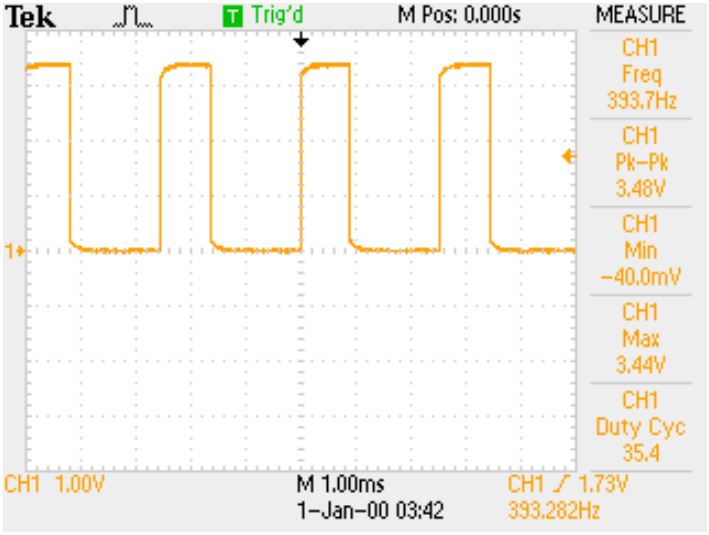


Figura 71: Señal medida en osciloscopio.

- Frecuencia de la señal de entrada: 393 Hz
- Diámetro de la rueda delantera izquierda: 167 cm = 1.67 m
- Numero de dientes del disco dentado: 23

$$Velocidad = \frac{393}{23} \cdot 1{,}67 = 28{,}53\ m/s = 102{,}7\ Km/h$$

Problema encontrado: Este problema fue un cúmulo de circunstancias... El generador de señales tenía presionado el botón de "20dB ATT" y la sonda del osciloscopio la tenía en "1X" en lugar de "10X" que era como estaba configurado el canal del osciloscopio, esto hizo que al ajustar la señal del generador de señales atendiendo a los valores que me mostraba el osciloscopio, la señal real fuera de 10 veces menos amplitud... las dos cosas juntas hicieron que no me diera cuenta en un principio y el microcontrolador no se enteraba de que le estaba inyectando una señal... Después del ajuste funcionó correctamente.

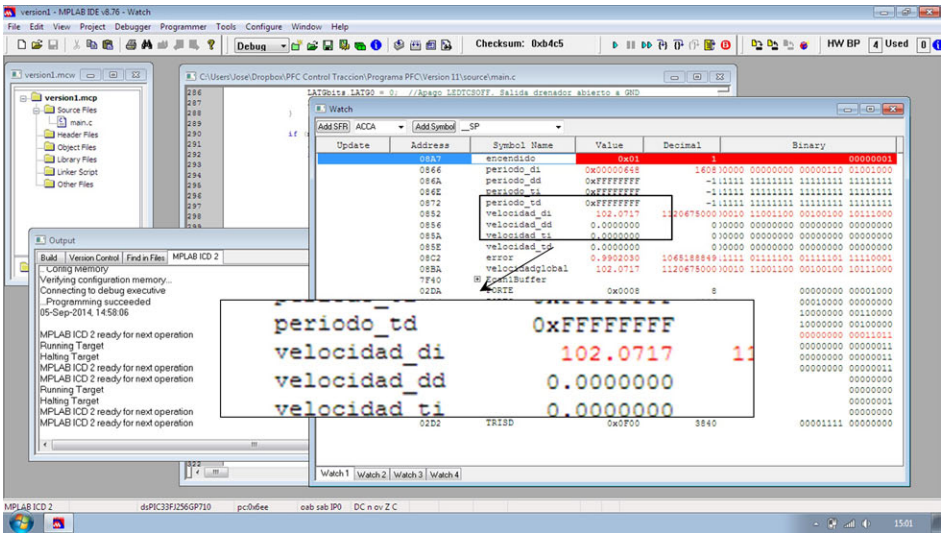


Figura 72: Captura de pantalla del debugger.

9.5. Señal de control de la centralita

Nuestro sistema de control de tracción genera una señal de control que ataca a la ECU del vehículo. Como la entrada de la ECU es digital, la señal generada por nuestro dispositivo debe tener dos estado: 0 voltios y 5 voltios, en otras palabras, la señal generada por el nuestro sistema es una señal cuadrada de 0 a 5 voltios de amplitud.

Para generarla hacemos uso del módulo PWM del microcontrolador y después la adaptamos con un circuito que convierte el nivel alto de 3,3 voltios de la salida del microcontrolador a 5 voltios.

Con el módulo PWM somos capaces de generar señales con diferente frecuencia y ciclo de trabajo. Para las pruebas, he modificado el programa para que cambie el ciclo de trabajo de la señal de salida según hacemos uso del selector en sus diferentes posiciones y así ver, además de su funcionamiento, su capacidad de variar este parámetro.

En las Figuras 73 y 74 se puede ver el resultado, tanto en la fotografía como en las imágenes capturadas desde los osciloscopios Tektronix de uno de los laboratorios de la escuela.

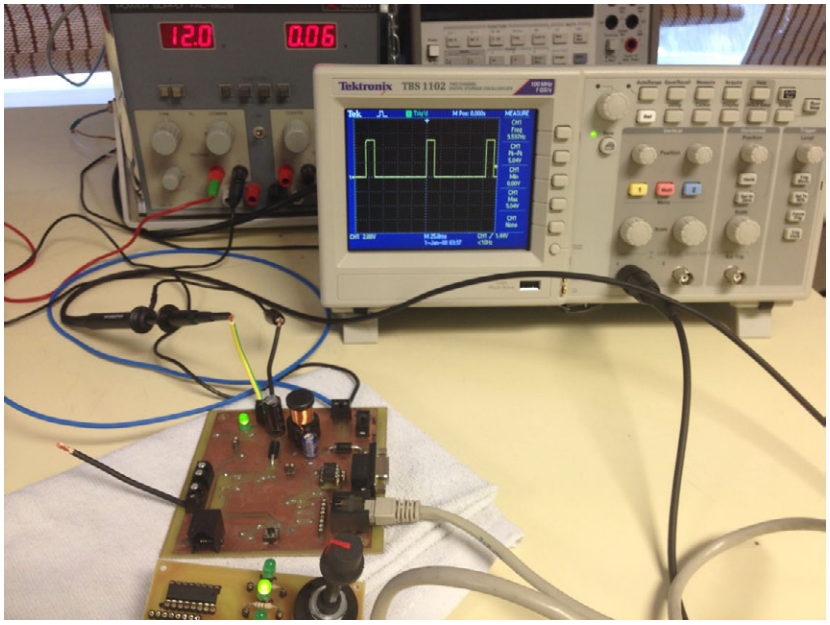


Figura 73: Señal medida con el osciloscopio.

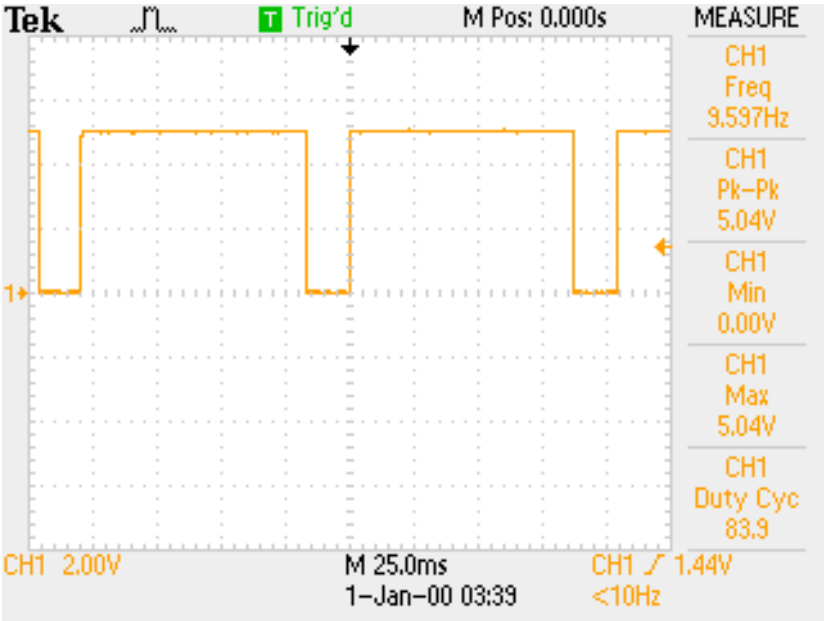


Figura 74: Señal medida con el osciloscopio.

9.6. Envío de una trama CAN

Para concluir con las pruebas realizadas a la placa, se analizó el envío de una trama de datos CAN. Para su análisis contamos con un kit de monitorización del bus CAN de la firma Microchip, concretamente el *MCP2515 CAN Bus Monitor Demo Board*.

Con este kit somos capaces de ver en el ordenador, gracias al software *MCP2515 Bus Monitor* de la propia Microchip, las tramas que enviadas desde el dispositivo de control de tracción. La placa de circuito impreso del kit de monitorización se conecta al ordenador mediante cable USB. La PCB del sistema de control de tracción y la PCB del kit de monitorización del bus CAN se conectan entre si con un cable con conectores DB9 macho.

Se ha configurado el software del dispositivo de control de tracción, para que envíe una trama de datos cuando ponemos el mando del selector del control remoto en posición "Mapa2", así controlamos el envío de tramas para las pruebas de esta parte del sistema.

La trama de datos enviada se configuró de la siguiente manera:

- Identificador de trama = 0x7F4.
- Byte 7 = 0x55
- Byte 6 = 0xFF
- Byte 5 = 0xFF
- Byte 4 = 0xFF
- Byte 3 = 0xFF
- Byte 2 = 0xFF

- Byte 1 = 0xFF
- Byte 0 = 0xFF

Comprobamos que el software *MCP2515 Bus Monitor* reflejaba correctamente la trama enviada desde el dispositivo de control de tracción. Para ello se tuvo que configurar el Baud Rate del bus CAN a 125 Kbit/s que ha sido la velocidad elegida. En la Figura 75 se muestra una captura de pantalla donde se refleja la trama enviada desde el sistema de control de tracción y recibida por el ordenador.

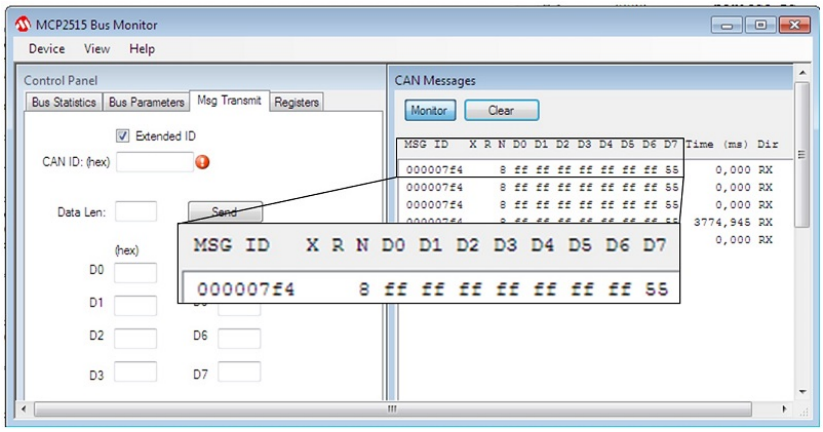


Figura 75: Trama de datos enviada vista desde el software *MCP2515 Bus Monitor*

También se consiguió ver la trama desde el osciloscopio y se comprobó, midiendo el tiempo de la señal a nivel alto de un bit, que el periodo de bit era de 8μs (lo que corresponde a un baud rate de 125 Kbit/s). En las Figuras 76 y 77 se aprecian las imagen tomadas de la pantalla del osciloscopio.

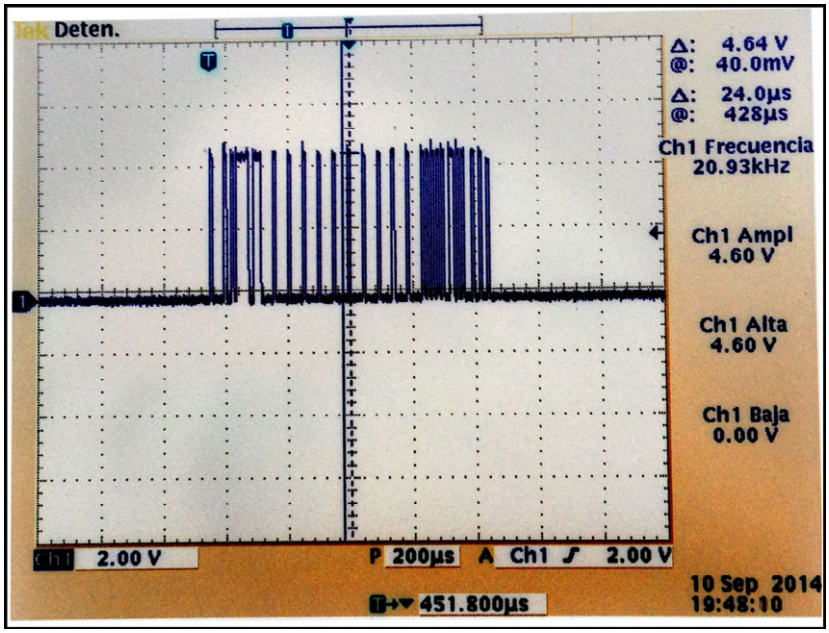


Figura 76: Trama de datos enviada vista desde el osciloscopio

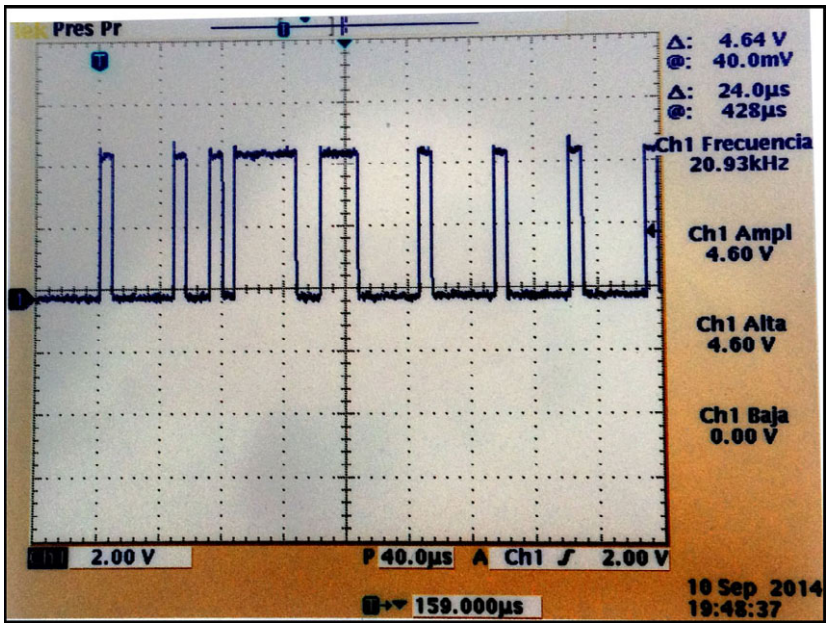


Figura 77: Trama de datos enviada vista desde el osciloscopio

10. Manual de uso y ajuste del sistema

10.1. Parámetros del sistema

El sistema cuenta con diversas variables en su funcionamiento. Aquellas que pueden ser modificados por el usuario son las siguientes:

- Longitud de la circunferencia de las ruedas delanteras
- Longitud de la circunferencia de las ruedas traseras
- Nº de dientes del disco dentado de la rueda
- Umbral a alta velocidad: Por encima de este valor se considera se circula a alta velocidad
- Umbral a baja velocidad: Por debajo de este valro se considera que se circula a baja velocidad
- Porcentaje de deslizamiento permitido a baja velocidad
- Porcentaje de deslizamiento permitido a media velocidad
- Porcentaje de deslizamiento permitido a alta velocidad
- Constante proporcional del regulador:

  Para un mismo modo de funcionamiento (de los tres mapas disponibles), se tienen 3 valores diferentes, uno por franja de velocidad.

  - Kp a baja velocidad MAPA1
  - Kp a media velocidad MAPA1
  - Kp a alta velocidad MAPA1
  - Kp a baja velocidad MAPA2

- Kp a media velocidad MAPA2
- Kp a alta velocidad MAPA2
- Kp a baja velocidad MAPA3
- Kp a media velocidad MAPA3
- Kp a alta velocidad MAPA3

- Constante de tiempo integral del regulador:

  Para un mismo modo de funcionamiento (de los tres mapas disponibles), se tienen 3 valores diferentes, uno por franja de velocidad.

  - Ti a baja velocidad MAPA1
  - Ti a media velocidad MAPA1
  - Ti a alta velocidad MAPA1
  - Ti a baja velocidad MAPA2
  - Ti a media velocidad MAPA2
  - Ti a alta velocidad MAPA2
  - Ti a baja velocidad MAPA3
  - Ti a media velocidad MAPA3
  - Ti a alta velocidad MAPA3

10.2. Modificación de los parámetros mediante Bus CAN

Haciendo uso de los mensajes CAN que se implementan en el sistema podremos conocer el valor actual de los parámetros anteriormente citados, a través de los "mensajes de estatus", y también modificarlos, mediante los "mensajes de configuración".

Se detallan dos tesituras de uso que se nos pueden presentar al utilizar el sistema:

Tesitura 1 Se desea saber cómo está configurado el sistema. Concretamente, cual es el valor actual de circunferencia de rueda delantera.

1. Mediante algún kit de monitorización de bus CAN o dispositivo que pueda generar tramas CAN, se comprondrá una trama de petición remota con el identificador de mensaje 0x7F0 (correspondiente al mensaje STATU0 que devuelve la información de la circunferencia de las ruedas).
2. Al enviar esta trama de petición remota, el sistema de control de tracción responderá con una trama de datos correspondiente al mensaje STATUS0, que con el sistema de monitorización del bus CAN se podrán leer y analizar. Concretamente los bytes 0 y 1 contienen los valores correspondientes a la longitud de la circunferencia de la rueda delantera.

Tesitura 2 Se quiere realizar un ajuste del sistema cambiando el valor del porcentaje de deslizamiento permitido cuando el coche va a alta velocidad.

1. En este caso, se debe componer una trama con el mensaje CONF1, que es el que guarda el valor que se desea modificar.
2. Para componer el mensaje CONF1 primero se debe conocer qué valores actuales tiene el sistema, por lo que se tendrán que solicitar los mensajes de STATUS necesarios tal y como se explica en la Tesitura de uso anterior.
3. Cuando se sepan dichos valores, se compondrá el mensaje CONF1 con nuestro kit de monitorización CAN, escribiendo el valor correspondiente al parámetro que se desea modificar y rellenando los demás datos con el mismo valor que ya tenían. Posteriormente enviaremos el mensaje por el bus CAN. No se debe olvidar poner correctamente el código de seguridad (byte 7 del campo de datos).
4. El dispositivo de control de tracción recibirá la trama y la procesará, modificando las variables anteriores por lo valores actuales contenidos en el mensaje CONF1.

En las siguientes líneas se realiza un ejemplo de composición de una trama que contenga el mensaje CONF0.

El campo de datos del mensaje CONF0 presenta la siguiente estructura:

- Mensaje CONF0:

  Identificador de mensaje: 0x7F4

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
CODSEG	-	-	DIENTES	CTPE	CTPD	CDPE	CDPD

Tabla 18: Contenido del campo data

  - Byte 7: Código de seguridad. Sirve para que si un nodo, por error o mala configuración, envía una trama con el identificador del mensaje CONF0, nuestro sistema no procese los datos y los almacene. Para ello, antes comprobará que el código de seguridad sea igual a un valor prefijado y si lo es, se procesará el dato contenido en el mensaje. El valor del código de seguridad es 0x55.
  - Byte 6: No se usa.
  - Byte 5: No se usa.
  - Byte 4: número de dientes del disco dentado. Valores posibles de 0 a 255 (unsigned char).
  - Byte 3: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
  - Byte 2: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).
  - Byte 1: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte entera. Valores de 0 a 255 (unsigned char).
  - Byte 0: Longitud de la circunferencia de la rueda trasera medida en centímetros. Solo la parte decimal. Valores de 0 a 99 (unsigned char).

Valor que se desea dar a los parámetros incluidos en el mensaje CONF0:

- Longitud de circunferencia de ruedas delanteras: 164,43 cm
- Longitud de circunferencia de ruedas traseras: 167,56 cm
- Número de dientes del disco dentado: 23

Con los datos anteriormente mostrados los valores de los bytes del campo “datos” del mensaje CONF0 (dados en notación decimal y en notación hexadecimal), serían los siguientes:

- Byte7(CODSEG) = 85 = 0x55
- Byte6(No se usa) = No importa el valor
- Byte5(No se usa) = No importa el calor
- Byte4(DIENTES) = 23 = 0x17
- Byte3(CTPE) = 167 = 0xA7
- Byte2(CTPD) = 56 = 0x38
- Byte1(CDPE) = 164 = 0xA4
- Byte0(CDPD) = 43 = 0x2B
- NOTA: El valor del campo del identificador estándar del mensaje CONF0 debe ser: 0x7F4

11. Presupuesto

Los costes globales de un proyecto incluyen los siguientes:

- Costes de ingeniería: Engloba el tiempo empleado por el ingeniero en realizar el proyecto.
- Costes en material: Coste los componentes electrónicos y material utilizado en la fabricación del dispositivo
- Costes en herramientas: Las herramientas utilizadas como los kits de desarrollo, instrumentos de medida,
- Costes en software: Precio de los programas utilizados para el desarrollo del código, generación de documentos, etc.

Para el cálculo de los costes de este proyecto no se han tenido en cuenta algunos de los gastos por considerarse gastos amortizables en el desarrollo de otros proyectos o simplemente gastos demasiado genéricos como para incluirlos. Estos gastos son los siguientes:

- Herramientas para la fabricación de la PCB: soldador, alicantes, insoladora, taladro de banco, etc.
- Instrumentación de medida: osciloscopio, multímetro, generador de señales, etc.
- Hardware informático: ordenador.

Sí se han tenido en cuenta los siguientes gastos:

- Costes de ingeniería: Tiempo de desarrollo neto: 8 meses a 25 horas semanales. Precio/hora: 15 €/h.

- Material:
  - Componentes electrónicos: 30,91 €. Al final de este apartado se desglosa el listado de componentes.
  - Placa de cobre fotosensible (12 €), estaño (4 €), flux (5 €), líquidos para la fabricación PCB (6 €).
- Software de desarrollo: Utilizado en versión educativa (0 €).
- Hardware: Serían los dos kits de desarrollo utilizados, el "Bus CAN Monitor demo board" (50 €) y el "MPLAC ICD2" (50 €).

Estos gastos se resumen en la siguiente tabla.

Descripción	Precio
Costes de ingeniería	12000 €
Componentes electrónicos	30,91 €
Material	27 €
Software	0 €
Hardware	100 €
TOTAL	12157,91 €

Desglose de los componentes electrónicos utilizados y su precio:

RESISTENCIAS					
Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
10	1	1206	740-9085	0,02	0,02
10K	12	1206	740-9110	0,015	0,18
470	1	1206	740-9120	0,015	0,015
100	2	1206	740-9079	0,019	0,038
100K	2	1206	679-1761	0,026	0,052
150	5	1206	223-2142	0,058	0,29
4K7	1	1206	223-2350	0,017	0,017
1K	2	1206	740-9088	0,019	0,038
15k	1	1206	721-9917	0,142	0,142
8200	1	1206	721-9894	0,138	0,138

CONDENSADORES

Tántalo

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
10uF	2	C / 6032	684-5127	0,518	1,036
22uF	1	C / 6032	684-5152	0,518	0,518

Cerámicos

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
100nF	12	1206	669-8515	0,058	0,696
10nF	1	1206	669-8410	0,05	0,05
1uF	1	1206	648-0733	0,093	0,093
22nF	1	1206	669-8432	0,046	0,046

Electrolíticos aluminio

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
100uF	1	Orificio pasante 5x11	228-6903	0,188	0,188
1000uF	1	Orificio pasante 5x15	711-1150	0,223	0,223

DIODOS

Schottky

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
1N5822	2	Orificio pasante 9,5mm	687-0877	0,126	0,252

LED

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
Verde	5	Orificio pasante 2,54mm	228-6004	0,088	0,44
Rojo	1	Orificio pasante 2,54mm	228-5972	0,118	0,118

BOBINAS

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
100uH	1	Orificio pasante 5mm	233-5409	1,93	1,93

REGULADORES

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
LM2576S-5.0	1	TO-263	533-3226	2,72	2,72
LM1117IMP-3.3	1	SOT-223	535-8635	0,928	0,928

TRANSCEIVER CAN

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
MCP2551	1	PDIP 8 pines	402-920	1,01	1,01

CONECTORES Y PUNTOS PARA TEST

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
RJ11	1	Orificio pasante	735-0282	0,76	0,76
RJ45	2	Orificio pasante	735-0295	1,7	3,4
Conectores	6	Orificio pasante	425-8720	0,292	1,752
Tira de zócalo	2	Orificio pasante	267-7400	0,698	1,396

INTERRUPTORES

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
Interruptor desliz.	1	Orificio pasante	711-8423	2,29	2,29
Interruptor giratorio	1	Orificio pasante	320-736	2	2
Mando potenc.	1		777-7362	0,422	0,422
Interruptor reset	1	Orificio pasante	718-2415	0,144	0,144

MICROCONTROLADOR

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
DSPIC33FJ128GP710	1	100pin TQFP	666-9646	6,78	6,78

INTEGRADOS

Valor	Cantidad	Encapsulado	Código RS	Precio Ud	Precio Total
Texas Instru. ULN2003AN	1	DIP-16	436-8451	0,5	0,5
Microchip MCP6541	1	SOT-23-5	669-6212	0,29	0,29

TOTAL 30,91 €

12. Conclusiones y desarrollo del sistema creado

Tengo que decir que ha sido un verdadero reto enfrentarme a la realización de este PFC. El estudio y diseño del sistema de control de tracción me ha supuesto un gran esfuerzo, solo comparable a la satisfacción de verlo realizado.

Aunque durante la realización del proyecto me he encontrado con muchos altibajos, he disfrutado mucho en su elaboración porque entre otras cosas, me apasiona el campo de la automoción y me apasiona el campo de la electrónica.

Personalmente este proyecto me ha brindado la oportunidad de volverme a encontrar con mi vocación y afición por la electrónica, aquella que me hizo aventurarme con tanta motivación a cursar los estudios de ingeniería técnica en telecomunicación en la especialidad de sistemas electrónicos. Esta vocación, de algún modo, se fue viendo delegada a un segundo plano a causa de los diferentes puestos de trabajo que he desempeñado y que no han tenido que ver concretamente con este ámbito. La realización de este PFC, ha sido por tanto, una oportunidad y un motivo al mismo tiempo, para retomar lazos en este terreno.

Dejando a un lado esta parte de la conclusión un tanto sentimental y ciñéndome al sistema realizado, tengo que decir que uno cae perfectamente en la cuenta de que la incursión de la electrónica en los sistemas del automóvil es un adelanto muy importante tanto en seguridad como en prestaciones de los vehículos.

El sistema ha funcionado correctamente según los criterios establecidos y además es un sistema flexible y ajustable, que puede tener un buen desempeño en el vehículo. Me siento satisfecho con el resultado.

La parte amarga, pero inevitable, de este apartado, es decir que el sistema no se ha podido probar realmente en el monoplaza de la Fórmula SAE desarrollado por el

equipo UPMRacing. Esto es así porque yo comencé este PFC en el año 2010 y unos años después, se puso fin a la colaboración entre el departamento de sistemas y control de la EUITT y el INSIA. De cualquier forma, en el acercamiento que he hecho hace pocas semanas con el coordinador del equipo UPMRacing para pedir consejo sobre los métodos de disminución de potencia del motor, me ha parecido que podían tener interés en el sistema y no descarto colaborar con ellos para ayudar a instalarlo en el coche, cosa que me gustaría bastante.

Otra conclusión positiva, aunque negativa al mismo tiempo, es que al haberme familiarizado con el sistema de control de tracción, se me han ocurrido a posteriori diversas formas de mejorarlo para optimizar su funcionamiento y entre ellas se me ocurren las siguientes líneas de desarrollo:

- Limitador de velocidad: Con un botón más en los mandos del piloto y algo de software podríamos implementar un limitador de velocidad, que limitara la velocidad máxima en la entrada a boxes, por ejemplo.
- Función de toma de datos. Guardando todos los datos de velocidad de cada rueda en cada instante durante el tiempo de funcionamiento del coche en un trayecto, podríamos saber la distancia recorrida, la velocidad en cada punto y con ayuda de diversos cálculos, se podría conocer si el coche ha efectuado algún giro, incluso de cuántos grados ha sido (siempre que no haya pérdidas de adherencia en el recorrido). Esto podría dar datos relevantes para el análisis de la carrera. Se podría mejorar a su vez con un sensor de giro de la dirección, otro del cambio de marchas, de consumo de combustible, etc.

Las siguientes mejoras suponen cambios importantes en la forma de implementar el control de tracción:

- Añadir un sensor para conocer la posición del cigüeñal y así conocer exactamente cuando comienzan y terminan los tiempos de admisión en los cilindros. Con esto

podemos eliminar la inyección de combustible en el tiempo completo de admisión y cambiar el método de reducción de potencia del motor. Eliminando la inyección de combustible completamente en el tiempo de admisión de un cilindro, no tendremos la desventaja que tenemos con el método actual, de sacar combustible sin quemar por el escape. Además conociendo la posición del cigüeñal podemos actuar selectivamente en un cilindro o más teniendo un control más suave y preciso.

- Cambiar la mariposa del acelerador por una electroválvula. Esto también daría lugar a un control muy suave y preciso del sistema. Cuando se detecte deslizamiento, tomaríamos el control de la electroválvula, y reduciríamos la aceleración, simulando, para hacernos una idea, que el piloto levanta el pie del acelerador. Con este sistema sería relativamente fácil incorporar una función de limitador de velocidad e incluso un control de crucero.

13. Bibliografía

Artículos de la Wikipedia en español:

[1] *Fórmula Student.*
http://es.wikipedia.org/wiki/Formula_Student

[2] *Energía gris.*
http://es.wikipedia.org/wiki/Energia_gris

[3] *Ingeniería de Control.*
http://es.wikipedia.org/wiki/Ingenieria_de_control

[4] *Regulación automática.*
http://es.wikipedia.org/wiki/Regulacion_automatica

[5] *Sistemas de Control.*
http://es.wikipedia.org/wiki/Sistema_de_control

[6] *Sistema dinámico.*
http://es.wikipedia.org/wiki/Sistema_dinamico

[7] *Proporcional Integral derivativo.*
http://es.wikipedia.org/wiki/Proporcional_integral_derivativo

[8] *Sensor de efecto Hall.*
http://es.wikipedia.org/wiki/Sensor_de_efecto_Hall

[9] *Efecto Hall.*
http://es.wikipedia.org/wiki/Efecto_Hall

[10] *Unidad de Control de motor.*
http://es.wikipedia.org/wiki/Unidad_de_control_de_motor

[11] *Microcontrolador.*
http://es.wikipedia.org/wiki/Microcontrolador

[12] *Control de tracción.*
http://es.wikipedia.org/wiki/Control_de_traccion

[13] *Control de estabilidad.*
http://es.wikipedia.org/wiki/Control_de_estabilidad

[14] *Bus CAN.*
http://es.wikipedia.org/wiki/Bus_CAN

Artículos de la Wikipedia en inglés:

[15] *Formula Student.*
http://en.wikipedia.org/wiki/Formula_Student

[16] *Fórmula SAE.*
http://en.wikipedia.org/wiki/Formula_SAE

[17] *SAE International.*
http://en.wikipedia.org/wiki/SAE_International

Información variada sobre Fórmula SAE:

[18] *Ranking mundial Formula Student.*
http://www.fs-world.org

[19] *Página web de la SAE International.*
http://www.sae.org

[20] *Página web del evento español de la Fórmula Student.*
http://www.formulastudent.es/

[21] *Web del equipo UMPRacing.*
http://www.upmracing.es

Otros

[22] Lasheras, Juan, *Diferenciales de deslizamiento limitado: El Arte y la Ciencia.* http://8000vueltas.com/

[23] *Diferenciales.* http://www.mecanicavirtual.org/diferencial-autoblocante.htm

[24] Costo Grimaney, Jorge, *Reguladores y Teoría Básica sobre Control.* Facultad de ingeniería química y textil, Universidad Nacional de Ingeniería de Lima, 2008

[25] *Data Sheets de los componentes. Descargados desde:.* http://es.rs-online.com

[26] *Data Sheets y manuales del microcontrolador DSPICFJ33.* http://www.microchip.com/pagehandler/en-us/family/16bit/architecture/dspic33f.html?f=4

[27] *Manual de la centralita PE-ECU-01 del monoplaza.* http://www.pro-1performance.com/

14. ANEXOS

14.1. Código del programa desarrollado

En las siguientes páginas se muestra el código del programa que se ha desarrollado para el microcontrolador del sistema.

En la versión *pdf* de este documento, la sintáxis se encuentra resaltada en colores para que resulte más fácil su lectura.

En el CD adjunto entregado con este libro, se encuentran todos los ficheros necesarios para abrir el proyecto en el entorno de desarrollo MPLAB IDE.

```
/*************************************************************************************************

Fichero:        main.c
Autor:          José Manuel García Villegas
Fecha:          24/06/2014
Versión:        V11

Descripción:

Programa para el control de tracción de un monoplaza de la Fórmula SAE.
Lee la señal proveniente de 4 sensores (uno situado en cada rueda del monoplaza)
y calcula la velocidad de cada rueda. Con esto analiza si existe deslizamiento y
si verdaderamente existe, genera una señal que se manda a la centralita del vehiculo
para que se disminuya la potencia del motor.

Histórico de versiones:

V11   24/06/2014    Mejora del BUS CAN.

V10   12/06/2014    Añadida la parte del BUS CAN. Se comprueba que el código compila sin errores.
                    Se crea fichero de estimulos para depurar y testear el código.
                    Se hace limpieza del código para mejorar la lectura.

V9    29/05/2014    Cambio en la función control(). Se han quitado las constantes proporcional y
                    tiempo integral de una de las franjas de velocidad. Y se ha añadido una
                    sentencia para que se resetee el error integral acumulado cuando el sistema
                    ya no presente error actual. (explicado dentro de la función)

V8    20/04/2014    Cambio en la funcion evaluar() para generar correctamente la señal de error
                    que representa el deslizamiento del sistema, porque antes no tenía en cuenta
                    dejar un cierto umbral de velocidad muy baja que podría alterar la señal
                    (explicado mejor dentro de la funcion)

V7    11/04/2014    Eliminación de variables que no usaba y de una de las franjas de
                    velocidad a tener en cuenta en la generación de la señal de control. Limpieza
                    del código.

V6    23/03/2014    Correcciones varias en la estructura de la funcion main

V5    14/03/2014    Nueva forma de calculo de la velocidad de las ruedas.
                    Ahora se mide el tiempo entre los pulsos de la señal y no
                    como antes que se contaba el nº de pulsos.
...

*************************************************************************************************/



#include <p33FJ256GP710.h>



//************************************************************
//     CONFIGURACIÓN DE LOS REGISTROS DEL MICROCONTROLADOR
//************************************************************

//NOTA: Oscilador, prioridades, etc.
//Selección del Oscilador Interno en el Power on Reset
_FOSCSEL(FNOSC_FRC);
//Habilitamos el cambio de oscilador por software y configuramos el pin OCS2 como salida para la
//señal FCY
_FOSC(FCKSM_CSECMD & OSCIOFNC_OFF);







//*****************************************
//     DEFINICIÓN DE CONSTANTES (#define)
//*****************************************

#define MSaKMH 3.6
#define MAPA1 1
#define MAPA2 2
```

```
#define MAPA3 3
#define PERIODOTIMER3 0.0000016 //Periodo para el timer asociado al Input Capture que servirá para
                                //medir el periodo de las señales de los sensores de las ruedas.




//****************************************
//     DEFINICIÓN DE VARIABLES GLOBALES
//****************************************

unsigned char ALTAVELOCIDAD = 80; //Umbrales de velocidad en km/h.
unsigned char BAJAVELOCIDAD = 20;

float velocidad_di = 0; //Valor la velocidad de giro de la rueda delantera izquierda
float velocidad_dd = 0; //Valor la velocidad de giro de la rueda delantera derecha
float velocidad_ti = 0; //Valor la velocidad de giro de la rueda trasera izquierda
float velocidad_td = 0; //Valor la velocidad de giro de la rueda trasera derecha

//Guardan el nº de desbordamientos en el timer3 referente a cada canal del Input Capture
unsigned char numdesbordamientos_IC1 = 0;
unsigned char numdesbordamientos_IC2 = 0;
unsigned char numdesbordamientos_IC3 = 0;
unsigned char numdesbordamientos_IC4 = 0;

//Variables que guardan el nº de incrementos del Timer3 entre cada pulso de las señales
//de los sensores de efecto Hall de cada rueda. Equivale al periodo de dichas señales.
long periodo_di = -1;
long periodo_dd = -1;
long periodo_ti = -1;
long periodo_td = -1;

//Variable que guarda el número de cuenta anterior que captura el módulo Input Capture para cada
//canal, con el fin de restarselo al nuevo valor capturado para obtener el periodo de la señal.
long bufferanterior_IC1 = -1;
long bufferanterior_IC2 = -1;
long bufferanterior_IC3 = -1;
long bufferanterior_IC4 = -1;

unsigned char erroraltavelocidad = 10; //Errores permitidos. En tanto por ciento.
unsigned char errormediavelocidad = 20;
unsigned char errorbajavelocidad = 80;

float circunferenciatrasera = 164.38; //Longitud de la circunferencia de las ruedas en cm
float circunferenciadelantera = 167.57;
unsigned char dientes = 23; //Numero de dientes del disco dentado acoplado a la rueda.

//Constantes para cambiar el comportamiento del regulador según la franja de velocidad en que
//nos movamos. Podemos necesitar un control menos "intrusivo" para salidas desde parado que para
//velocidades altas en los que un deslizamiento es un peligro para la seguridad del piloto.
//Existe un valor de constante proporcional y de tiempo integral por cada franja de velocidad
//a tener en cuenta: baja(b), media(m) y alta(a).
unsigned char Kpb1=10,Kpm1=10,Kpa1=10,Tib1=10,Tim1=10,Tia1=10; //constantes del regulador MAPA 1
unsigned char Kpb2=10,Kpm2=10,Kpa2=10,Tib2=10,Tim2=10,Tia2=10; //constantes del regulador MAPA 2
unsigned char Kpb3=10,Kpm3=10,Kpa3=10,Tib3=10,Tim3=10,Tia3=10; //constantes del regulador MAPA 3
//IMPORTANTE: Las constantes se guardan multiplicadas por 10, para facilitar su envío y
//adaptación a la hora de componer el mensaje enviado por el BUS CAN y facilitar también
//su almacenamiento como unsigned char (en lugar de float)

unsigned char datospreparados = 0; //Cuando esta variable está a 1, se ejecuta el sistema 1 vez.
                                   //Nos permite definir el periodo de análisis del sistema.
unsigned char prepararmsgcan = 1; //Variable que controla la composición de los mensajes del BUSCAN

float erroracumulado; //Variable que guarda los datos del error acumulado para el correcto cálculo
                      //de la acción integral del regulador PI que estamos simulando.

unsigned char manteneraccion = 0; //Variable que sirve para que la funcion del controlador sepa
                                  //que tiene que seguir aplicando señal de control aunque el
                                  //error ya se haya hecho cero.

unsigned char encendido=0;

static unsigned int Ecan1Buffer[12][8] __attribute__((space(dma))); //asignamos espacio para 12
                                                                    //buffers de 8 palabras en la
                                                                    //DMA RAM
```

```
//*******************************************
//     DEFINICIÓN DE CABECERA DE FUNCIONES
//*******************************************

void inicializacion_T1(void);
void inicializacion_T3(void);
void inicializacion_IC1_IC2_IC3_IC4(void);
void inicializacion_T2_PWM(void);
void inicializacion_ECAN1(void);
void inicializacion_DMA(void);
void inicializacion_puertos(void);

void composicion_msgcan(void);

float calcularvelocidadglobal(float dd1, float di1);
float evaluar(float dd1, float di1, float td1, float ti1, float velocidad);
void devolverconstantes (float velocidad, unsigned char mapa, float *Kp, float *Ti);
float control (float,float,float,float);
void actuar(float);









//************************
//     PROGRAMA MAIN
//************************

int main(void){

    //Configuración del oscilador para conseguir 40MIPS con el Oscilador Interno + PLL

    //Configuramos el PLLPRE, PLLPOST y PLLDIV con los valores adecuados
    PLLFBD = 41;            // M = 43
    CLKDIVbits.PLLPOST=0;   // N2 = 2
    CLKDIVbits.PLLPRE=0;    // N1 = 2

    //Iniciamos el cambio de oscilador para activar el Oscilador Interno + PLL (NOSC = 0b001)
    __builtin_write_OSCCONH(0x01);
    __builtin_write_OSCCONL(0x01);
    // NOTA: Estas son unas instrucciones especiales porque para escribir en el registro OSCCON
    // es necesario hacer un proceso de desbloqueo especial (Section 7 - Oscilator, pagina 31)

    //Esperamos a que se ejecute el cambio de oscilador
    while (OSCCONbits.COSC != 0b001) {};

    //Esperamos a que se bloquee el PLL. Esto nos indica que ya está en funcionamiento
    while (OSCCONbits.LOCK != 1) {};


    //Variables locales
    float error=0; //variable que guarda el error actual del sistema.
    float Kp=1; //ganancia de la acción proporcional actual
    float Ti=1; //valor del tiempo integral actual
    float salida=0; //
    float velocidadglobal=0; //variable que guarda la velocidad del sistema en cada iteración.
    unsigned char mapa = 1;
    unsigned int puertog = 0;
    unsigned char ledtcsoff = 0;
    unsigned char ledmapa1 = 0;
    unsigned char ledmapa2 = 0;
    unsigned char ledmapa3 = 0;


    //Inicialización de periféricos

    inicializacion_T1(); //Timer1: Genera el periodo de analisis del sistema
```

```
inicializacion_T3(); //Timer3: Tiempo base para el modulo Inputo Capture
inicializacion_IC1_IC2_IC3_IC4(); //Canales del módulo Input Capture.
                                  //Miden la velocidad de las ruedas
inicializacion_T2_PWM();
inicializacion_puertos();
inicializacion_ECAN1(); //Configuración del modulo ECAN1 para el control del BUS CAN
inicializacion_DMA(); //Configuración de la DMA para transferir los mensajes del ECAN1

T2CONbits.TON = 1; //Encendemos el Timer2 para que comience el PWM

while(1){

    //Leemos los pines RE3 y RE4 para establecer si es sistema está encendido y guardar
    //el mapa de motor seleccionado.
    encendido = PORTE;
    encendido = encendido & 0b0000000000011000;
    encendido = encendido >> 3;

    if (encendido == 1){
        mapa = MAPA2;
    }
    else if (encendido == 2){
        mapa = MAPA1;
    }
    else if (encendido == 3){
        mapa = MAPA3;
    }

    //Leemos el puerto G para comprobar el estado de los leds y guardar su valor en variables
    puertog = LATG;

    ledmapa3 = (puertog & 0b0010000000000000) >> 13; //pin RG13
    ledmapa2 = (puertog & 0b0001000000000000) >> 12; //pin RG12
    ledmapa1 = (puertog & 0b0100000000000000) >> 14; //pin RG14
    ledtcsoff = (puertog & 0b0000000000000001); //pin RG0

    //Código del BUS CAN: Composición de las variables a enviar referentes al BUS CAN.
    //La variable "prepararmsgcan" es una variable global que se pondrá a 1 en la RSI
    //de la DMA cuando se reciba un mensaje válido desde el BUS CAN, lo que implicará que
    //actualicemos los datos de los mensajes de STATUS.
    if (prepararmsgcan == 1){
        composicion_msgcan();
        prepararmsgcan = 0;
    }


    if(encendido){ //Acciones a llevar a cabo cuando el sistema está ON

        //Si LEDTCSOFF está encendido, lo apago.
        if (ledtcsoff == 1){
            LATGbits.LATG0 = 0;  //Apago LEDTCSOFF. Salida drenador abierto a GND
            ledtcsoff = 0;
        }

        if (mapa == MAPA1){
            //Si LEDMAPA1 está apagado, lo enciendo y apago los demás.
            if (ledmapa1 == 0){
                LATGbits.LATG14 = 1 ;  //LEDMAPA1
                LATGbits.LATG12 = 0;  //LEDMAPA2
                LATGbits.LATG13 = 0;   //LEDMAPA3
                ledmapa1 = 1;
                ledmapa2 = 0;
                ledmapa3 = 0;
                OC1RS = 2000; //Como prueba!: genera un PWM con ciclo de trabajo muy bajo.
            }
        }//fin del if

        else if (mapa == MAPA2){
            //Si LEDMAPA2 está apagado, lo enciendo y apago los demás.
            if (ledmapa2 == 0){
                LATGbits.LATG14 = 0 ;  //LEDMAPA1
                LATGbits.LATG12 = 1;  //LEDMAPA2
                LATGbits.LATG13 = 0;   //LEDMAPA3
                ledmapa1 = 0;
                ledmapa2 = 1;
                ledmapa3 = 0;
                OC1RS = 9000; //Como prueba!: genera un salida PWM con ciclo de trabajo bajo
            }
        }//fin del else if
```

```
    else if (mapa == MAPA3){
        //Si LEDMAPA3 está apagado, lo enciendo y apago los demás.
        if (ledmapa3 == 0){
            LATGbits.LATG14 = 0 ;  //LEDMAPA1
            LATGbits.LATG12 = 0;  //LEDMAPA2
            LATGbits.LATG13 = 1;   //LEDMAPA3
            ledmapa1 = 0;
            ledmapa2 = 0;
            ledmapa3 = 1;
            OC1RS = 55000; //Como prueba!: genera una salida PWM con ciclo de trabajo alto
        }
    }//fin del else if


    if(T1CONbits.TON == 0){ //si el timer1 está apagado hay que encenderlo
        TMR1 = 0x0000; //Inicializamos valor de cuenta del timer1
        T1CONbits.TON = 1; //Encendemos el timer1. Comenzará su cuenta
    }//fin del if

    if(T3CONbits.TON == 0){ //si el timer3 está apagado hay que encenderlo
        TMR3 = 0x0000; //Inicializamos valor de cuenta del timer3
        T3CONbits.TON = 1; //Encendemos el timer3. Comenzará su cuenta
    }//fin del if

    if(IC1CONbits.ICM == 0b000){ //si el canal IC1 está deshabilitado hay que habilitarlo
        IC1CONbits.ICM = 0b010; //Habilitamos el modulo IC1
    }//fin del if

    if(IC2CONbits.ICM == 0b000){ //si el canal IC2 está deshabilitado hay que habilitarlo
        IC2CONbits.ICM = 0b010; //Habilitamos el modulo IC2
    }//fin del if

    if(IC3CONbits.ICM == 0b000){ //si el canal IC3 está deshabilitado hay que habilitarlo
        IC3CONbits.ICM = 0b010; //Habilitamos el modulo IC3
    }//fin del if

    if(IC4CONbits.ICM == 0b000){ //si el canal IC4 está deshabilitado hay que habilitarlo
        IC4CONbits.ICM = 0b010; //Habilitamos el modulo IC4
    }//fin del if

    if(datospreparados){
        velocidadglobal = calcularvelocidadglobal(velocidad_dd,velocidad_di);
        error=evaluar(velocidad_dd,velocidad_di,velocidad_td,velocidad_ti,velocidadglobal);
        devolverconstantes(velocidadglobal,mapa,&Kp,&Ti);
        salida=control(error,Kp,Ti,velocidadglobal);
        actuar(salida);

        datospreparados=0;
    }//fin del if

}//fin del if

else{ //Acciones a llevar a cabo cuando el sistema se pone en OFF

    //Si LEDTCSOFF está apagado, lo enciendo y apago los leds de los mapas
    if (ledtcsoff == 0){
        LATGbits.LATG0 = 1;  //LEDTCSOFF. Salida drenador abierto en alta impedancia
        LATGbits.LATG14 = 0 ;  //LEDMAPA1. Salida drenador abierto a GND
        LATGbits.LATG12 = 0;  //LEDMAPA2. Salida drenador abierto a GND
        LATGbits.LATG13 = 0;   //LEDMAPA3. Salida drenador abierto a GND
        ledtcsoff = 1;
        ledmapa1 = 0;
        ledmapa2 = 0;
        ledmapa3 = 0;
    }

    if(T1CONbits.TON == 1){ //si el timer1 está encendido hay que apagarlo
        T1CONbits.TON = 0; //Apagamos el Timer1
    }//fin del if

    if(T3CONbits.TON == 1){ //si el timer3 está encendido hay que apagarlo
        T3CONbits.TON = 0; //Apaganos el Timer3
    }//fin del if

    if(IC1CONbits.ICM == 0b010){ //si el canal IC1 está habilitado hay que deshabilitarlo
        IC1CONbits.ICM = 0b000; //Deshabilitamos el modulo IC1
    }//fin del if

    if(IC2CONbits.ICM == 0b010){ //si el canal IC2 está habilitado hay que deshabilitarlo
        IC2CONbits.ICM = 0b000; //Deshabilitamos el modulo IC2
```

```
            }//fin del if

            if(IC3CONbits.ICM == 0b010){ //si el canal IC3 está habilitado hay que deshabilitarlo
                IC3CONbits.ICM = 0b000; //Deshabilitamos el modulo IC3
            }//fin del if

            if(IC4CONbits.ICM == 0b010){ //si el canal IC4 está habilitado hay que deshabilitarlo
                IC4CONbits.ICM = 0b000; //Deshabilitamos el modulo IC4
            }//fin del if

            //T2CONbits.TON = 0; //Parar timer2-PWM NO SE PARA EL TIMER 2

            if(OC1RS != 30000){ //registro del Output Compare que genera la señal PWM
                OC1RS = 30000; //Aseguramos que el PWM tenga salida = 0 voltios.
            }//fin del if

            salida=0;

            datospreparados = 0; //dejamos preparado el sistema para la proxima vez

            erroracumulado = 0; //Inicializamos los errores de iteraciones pasadas

        }//fin del else

    }//fin del while

}//fin del programa main







//*****************************************************************
//     DEFINICIÓN DE RUTINAS DE SERVICIO DE LAS INTERRUPCIONES
//*****************************************************************

//-------------------------------------------------------
//     Inicio de la RSI del Timer 1
//-------------------------------------------------------

void __attribute__((__interrupt__)) _T1Interrupt(void){

    if (encendido){ //si el sistema está encendido

        if(datospreparados==0){

            //Calcular velocidad de las ruedas y guardarlas en sus variables correspondientes

            //Rueda Delantera Izquierda
            if (periodo_di == -1){
                velocidad_di = 0;
            }
            else {
                velocidad_di = (((circunferenciadelantera/100) / (float)dientes) * MSaKMH) /
                               ((float)periodo_di * PERIODOTIMER3);
            }

            //Rueda Delantera Derecha
            if (periodo_dd == -1){
                velocidad_dd = 0;
            }
            else {
                velocidad_dd = (circunferenciadelantera/100 / (float)dientes) * MSaKMH /
                               ((float)periodo_dd * PERIODOTIMER3);
            }

            //Rueda Trasera Izquierda
            if (periodo_ti == -1){
                velocidad_ti = 0;
            }
            else {
                velocidad_ti = (circunferenciatrasera/100 / (float)dientes) * MSaKMH /
                               ((float)periodo_ti * PERIODOTIMER3);
            }

            //Rueda Trasera Derecha
```

```
            if (periodo_td == -1){
                velocidad_td = 0;
            }
            else {
                velocidad_td = (circunferenciatrasera/100 / (float)dientes) * MSaKMH /
                               ((float)periodo_td * PERIODOTIMER3);
            }

            datospreparados = 1; //Activamos variable de datos preparados

        }//fin del if

                    else{
                                //ha habido solape en los datos;
                    }//fin del else

          }//fin del if

          else {

        //Si el sistema está apagado no hacemos nada

          }//fin del else

          IFS0bits.T1IF = 0; //Importante borrar el flag de la interrupción.

}//Fin de la RSI del TImer1

//-------------------------------------------------------
//     Fin de la RSI del Timer1
//-------------------------------------------------------




//-------------------------------------------------------
//    Inicio de la RSI del Timer2-PWM
//-------------------------------------------------------

void __attribute__((__interrupt__)) _T2Interrupt(void){

          IFS0bits.T2IF = 0; //Importante borrar el flag de la interrupción.

}//Fin de la RSI del Timer2-PWM

//-------------------------------------------------------
//     Fin de la RSI del Timer2-PWM
//-------------------------------------------------------




//-------------------------------------------------------
//     Inicio de la RSI del Timer3
//-------------------------------------------------------

void __attribute__((__interrupt__)) _T3Interrupt(void){

    numdesbordamientos_IC1++;
    numdesbordamientos_IC2++;
    numdesbordamientos_IC3++;
    numdesbordamientos_IC4++;

    if (numdesbordamientos_IC1 > 2){
        periodo_di = -1;
        numdesbordamientos_IC1 = 3;
    }

    if (numdesbordamientos_IC2 > 2){
        periodo_dd = -1;
        numdesbordamientos_IC2 = 3;
    }

    if (numdesbordamientos_IC3 > 2){
        periodo_ti = -1;
        numdesbordamientos_IC3 = 3;
    }

    if (numdesbordamientos_IC4 > 2){
        periodo_td = -1;
```

```
        numdesbordamientos_IC4 = 3;
    }

    IFS0bits.T3IF = 0; //Borrar flag de la interrupción del Timer3

}//Fin de la RSI del Timer3

//-----------------------------------------------------
//     Fin de la RSI del Timer3
//-----------------------------------------------------




//-----------------------------------------------------
//     Inicio de la RSI del Input Capture 1
//-----------------------------------------------------

void __attribute__((__interrupt__)) _IC1Interrupt(void){

    //Rutina de atención a la interrupcion del canal IC1, que es el que está asociado
    //a la señal de la Rueda Delantera Izquierda.

    unsigned int bufferactual = 0;

    bufferactual = IC1BUF; //Leemos el valor del buffer del IC1

    if (bufferanterior_IC1 != -1){

        if (numdesbordamientos_IC1 == 0){
            if (bufferanterior_IC1 == PR3){
                periodo_di = (long)bufferactual + 1;
            }
            else {
                periodo_di = (long)bufferactual - bufferanterior_IC1;
            }
        }//fin del if

        else if (numdesbordamientos_IC1 >= 1 && numdesbordamientos_IC1 <= 2){
            if (bufferanterior_IC1 == bufferactual){
                periodo_di = numdesbordamientos_IC1 *((long)PR3 + 1);
            }
            else if (bufferanterior_IC1 == PR3){
                periodo_di = (numdesbordamientos_IC1 *((long)PR3 + 1)) + bufferactual + 1;
            }
            else if (bufferactual == PR3){
                periodo_di = (numdesbordamientos_IC1 *(long)PR3) - bufferanterior_IC1;
            }
            else {
                periodo_di = (numdesbordamientos_IC1 *((long)PR3 + 1)) -
                                    bufferanterior_IC1 + (bufferactual);
            }
        }//fin del else if

        else {
            periodo_di = -1;
        }

    }//fin del if

    bufferanterior_IC1 = bufferactual;
    numdesbordamientos_IC1 = 0; //Reiniciamos el valor de los desbordamientos de cuenta del Timer3
    IFS0bits.IC1IF = 0; //Borramos el flag de la interrupción del IC1


}//Fin de la RSI del IC1

//-----------------------------------------------------
//     Fin de la RSI del Input Capture 1
//-----------------------------------------------------




//-----------------------------------------------------
//     Inicio de la RSI del Input Capture 2
//-----------------------------------------------------

void __attribute__((__interrupt__)) _IC2Interrupt(void){
```

```
    //Rutina de atención a la interrupcion del canal IC2, que es el que está asociado
    //a la señal de la Rueda Delantera Derecha.

    unsigned int bufferactual = 0;

    bufferactual = IC2BUF; //Leemos el valor del buffer del IC2

    if (bufferanterior_IC2 != -1){

        if (numdesbordamientos_IC2 == 0){
            if (bufferanterior_IC2 == PR3){
                periodo_dd = (long)bufferactual + 1;
            }
            else {
                periodo_dd = (long)bufferactual - bufferanterior_IC2;
            }
        }//fin del if

        else if (numdesbordamientos_IC2 >= 1 && numdesbordamientos_IC2 <= 2){
            if (bufferanterior_IC2 == bufferactual){
                periodo_dd = numdesbordamientos_IC2 *((long)PR3 + 1);
            }
            else if (bufferanterior_IC2 == PR3){
                periodo_dd = (numdesbordamientos_IC2 *((long)PR3 + 1)) + bufferactual + 1;
            }
            else if (bufferactual == PR3){
                periodo_dd = (numdesbordamientos_IC2 *(long)PR3) - bufferanterior_IC2;
            }
            else {
                periodo_dd = (numdesbordamientos_IC2 *((long)PR3 + 1)) -
                                bufferanterior_IC2 + (bufferactual);
            }
        }//fin del else if

        else {
            periodo_dd = -1;
        }

    }//fin del if

    bufferanterior_IC2 = bufferactual;
    numdesbordamientos_IC2 = 0; //Reiniciamos el valor de los desbordamientos de cuenta del Timer3
    IFS0bits.IC2IF = 0; //Borramos el flag de la interrupción del IC2


}//Fin de la RSI del IC2

//-------------------------------------------------------
//     Fin de la RSI del Input Capture 2
//-------------------------------------------------------




//-------------------------------------------------------
//     Inicio de la RSI del Input Capture 3
//-------------------------------------------------------

void __attribute__((__interrupt__)) _IC3Interrupt(void){

    //Rutina de atención a la interrupcion del canal IC3, que es el que está asociado
    //a la señal de la Rueda Trasera Izquierda.

    unsigned int bufferactual = 0;

    bufferactual = IC3BUF; //Leemos el valor del buffer del IC3

    if (bufferanterior_IC3 != -1){

        if (numdesbordamientos_IC3 == 0){
            if (bufferanterior_IC3 == PR3){
                periodo_ti = (long)bufferactual + 1;
            }
            else {
                periodo_ti = (long)bufferactual - bufferanterior_IC3;
            }
        }//fin del if

        else if (numdesbordamientos_IC3 >= 1 && numdesbordamientos_IC3 <= 2){
            if (bufferanterior_IC3 == bufferactual){
```

```
                periodo_ti = numdesbordamientos_IC3 *((long)PR3 + 1);
            }
            else if (bufferanterior_IC3 == PR3){
                periodo_ti = (numdesbordamientos_IC3 *((long)PR3 + 1)) + bufferactual + 1;
            }
            else if (bufferactual == PR3){
                periodo_ti = (numdesbordamientos_IC3 *(long)PR3) - bufferanterior_IC3;
            }
            else {
                periodo_ti = (numdesbordamientos_IC3 *((long)PR3 + 1)) -
                                bufferanterior_IC3 + (bufferactual);
            }
        }//fin del else if

        else {
            periodo_ti = -1;
        }

    }//fin del if

    bufferanterior_IC3 = bufferactual;
    numdesbordamientos_IC3 = 0; //Reiniciamos el valor de los desbordamientos de cuenta del Timer3
    IFS2bits.IC3IF = 0; //Borramos el flag de la interrupción del IC3


}//Fin de la RSI del IC3

//-------------------------------------------------------
//     Fin de la RSI del Input Capture 3
//-------------------------------------------------------




//-------------------------------------------------------
//     Inicio de la RSI del Input Capture 4
//-------------------------------------------------------

void __attribute__((__interrupt__)) _IC4Interrupt(void){

    //Rutina de atención a la interrupcion del canal IC4, que es el que está asociado
    //a la señal de la Rueda Trasera Derecha.

    unsigned int bufferactual = 0;

    bufferactual = IC4BUF; //Leemos el valor del buffer del IC4

    if (bufferanterior_IC4 != -1){

        if (numdesbordamientos_IC4 == 0){
            if (bufferanterior_IC4 == PR3){
                periodo_td = (long)bufferactual + 1;
            }
            else {
                periodo_td = (long)bufferactual - bufferanterior_IC4;
            }
        }//fin del if

        else if (numdesbordamientos_IC4 >= 1 && numdesbordamientos_IC4 <= 2){
            if (bufferanterior_IC4 == bufferactual){
                periodo_td = numdesbordamientos_IC4 *((long)PR3 + 1);
            }
            else if (bufferanterior_IC4 == PR3){
                periodo_td = (numdesbordamientos_IC4 *((long)PR3 + 1)) + bufferactual + 1;
            }
            else if (bufferactual == PR3){
                periodo_td = (numdesbordamientos_IC4 *(long)PR3) - bufferanterior_IC4;
            }
            else {
                periodo_td = (numdesbordamientos_IC4 *((long)PR3 + 1)) -
                                bufferanterior_IC4 + (bufferactual);
            }
        }//fin del else if

        else {
            periodo_td = -1;
        }

    }//fin del if
```

```
    bufferanterior_IC4 = bufferactual;
    numdesbordamientos_IC4 = 0; //Reiniciamos el valor de los desbordamientos de cuenta del Timer3
    IFS2bits.IC4IF = 0; //Borramos el flag de la interrupción del IC4


}//Fin de la RSI del IC4

//-----------------------------------------------------
//     Fin de la RSI del Input Capture 4
//-----------------------------------------------------




//-----------------------------------------------------
//     Inicio de la RSI del Canal 0 de la DMA (RX)
//-----------------------------------------------------

void __attribute__((__interrupt__)) _DMA0Interrupt(void){

    //Hay que procesar el mensaje que ha llegado y que se ha guardado en el buffer correspondiente
    //de la DMA RAM.
    //Hay que sondear claramente de dónde ha venido la interrupción y en qué buffer se ha guardado
         //el mensaje.


    //Sondear motivo de la interrupción: Para ello usaremos los bit ICODE del registro C1VEC
    //de donde leeremos un código que nos dirá cual ha sido la fuente de interrupción.
    //En un funcionamiento normal, nuestro dispositivo solo generará interrupciones en la DMA0
    //cuando se reciban los mensajes CONF0, CONF1, CONF2, CONF3 y CONF4 que se almacenan en los
    //buffer 4, 5, 6, 7 y 8 respectivamente.

    unsigned char msgtemp1 = 0;
    unsigned char msgtemp2 = 0;

    //Para leer el registro C1VEC, el bit WIN tiene que estar a 0 (C1CTRL1bits.WIN = 0)
    C1CTRL1bits.WIN = 0;

    switch(C1VECbits.ICODE){

        case 0x00: //Interrupción por buffer 0
            break;

        case 0x01: //Interrupción por buffer 1
            break;

        case 0x02: //Interrupción por buffer 2
            break;

        case 0x03: //Interrupción por buffer 3
            break;

        case 0x04: //Interrupción por buffer 4
            //Ha llegado el mensaje CONF0 --> SID = 0x7F4
            //Es necesario comprobar el código de seguridad y en su caso guardar los datos.
            //El código de seguridad es un valor almacenado en el byte 7 del Payload del mensaje
            //CAN, con valor 0x55. Lo usamos para asegurarnos de que el mensaje es válido y así
            //proceder a guardar su contenido en las variables concretas del programa.

            //CONF0 palabra 3 = byte0 (Parte Decimal Circ. Rueda Delantera) +
                              //byte1 (Parte Entera Circ. Rueda Delantera)
                  //palabra 4 = byte2 (Parte Decimal Circ. Rueda Trasera) +
                              //byte3 (Parte Entera Circ. Rueda Trasera)
                  //palabra 5 = byte4 (N° de Dientes) +
                              //byte5 (Vacío)
                  //palabra 6 = byte6 (Vacío) +
                              //byte7 (Código de Seguridad = 0x55)

            if ( ((Ecan1Buffer[4][6] & 0xFF00) >> 8) == 0x55 ){

                msgtemp1 = Ecan1Buffer[4][3] & 0x00FF; //byte0 = P. Decimal Circunf. Rueda Del.
                msgtemp2 = (Ecan1Buffer[4][3] & 0xFF00) >> 8; //byte1 = P. Entera Circ. Rueda Del.
                circunferenciadelantera = msgtemp2 + ((float)msgtemp1 / 100);

                msgtemp1 = Ecan1Buffer[4][4] & 0x00FF; //byte2 = P. Decimal Circunf. Rueda Tr.
                msgtemp2 = (Ecan1Buffer[4][4] & 0xFF00) >> 8; //byte3 = P. Entera Circ. Rueda Tr.
                circunferenciatrasera = msgtemp2 + ((float)msgtemp1 / 100);

                dientes = Ecan1Buffer[4][5] & 0x00FF; //byte4 = N°Dientes
```

```
    }//fin del if

    break;

case 0x05: //Interrupción por buffer 5
    //Ha llegado el mensaje CONF1 --> SID = 0x7F5

    //CONF1 palabra 3 = byte0 (%error permitido a baja velocidad) +
                        //byte1 (%error permitido a media velocidad)
          //palabra 4 = byte2 (%error permitido a alta velocidad) +
                        //byte3 (Umbral Baja Velocidad)
          //palabra 5 = byte4 (Umbral Alta Velocidad) +
                        //byte5 (Vacío)
          //palabra 6 = byte6 (Vacío) +
                        //byte7 (Código de Seguridad = 0x55)

    if ( ((Ecan1Buffer[5][6] & 0xFF00) >> 8) == 0x55 ){

        errorbajavelocidad = Ecan1Buffer[5][3] & 0x00FF; //byte0 = %error perm. a baja vel.
        errormediavelocidad = (Ecan1Buffer[5][3] & 0xFF00) >> 8; //byte1 = %error perm. a
                                                                 //media velocidad
        erroraltavelocidad = Ecan1Buffer[5][4] & 0x00FF; //byte2 = %error perm. a alta vel.
        BAJAVELOCIDAD = (Ecan1Buffer[5][4] & 0xFF00) >> 8; //byte3 = Umbral baja vel.
        ALTAVELOCIDAD = Ecan1Buffer[5][5] & 0x00FF; //byte4 = Umbral alta vel.

    }//fin del if

    break;

case 0x06: //Interrupción por buffer 6
    //Ha llegado el mensaje CONF2 --> SID = 0x7F6

    //CONF2 palabra 3 = byte0 (Kpb1) +
                        //byte1 (Kpm1)
          //palabra 4 = byte2 (Kpa1) +
                        //byte3 (Tib1)
          //palabra 5 = byte4 (Tim1) +
                        //byte5 (Tia1)
          //palabra 6 = byte6 (Vacío) +
                        //byte7 (Código de Seguridad = 0x55)

    if ( ((Ecan1Buffer[6][6] & 0xFF00) >> 8) == 0x55 ){

        Kpb1 = Ecan1Buffer[6][3] & 0x00FF;         //byte0 = Kpb1
        Kpm1 = (Ecan1Buffer[6][3] & 0xFF00) >> 8; //byte1 = Kpm1
        Kpa1 = Ecan1Buffer[6][4] & 0x00FF;         //byte2 = Kpa1
        Tib1 = (Ecan1Buffer[6][4] & 0xFF00) >> 8; //byte3 = Tib1
        Tim1 = Ecan1Buffer[6][5] & 0x00FF;         //byte4 = Tim1
        Tia1 = (Ecan1Buffer[6][5] & 0xFF00) >> 8; //byte5 = Tia1

    }//fin del if

    break;

case 0x07: //Interrupción por buffer 7
    //Ha llegado el mensaje CONF3 --> SID = 0x7F7

    //CONF3 palabra 3 = byte0 (Kpb2) +
                        //byte1 (Kpm2)
          //palabra 4 = byte2 (Kpa2) +
                        //byte3 (Tib2)
          //palabra 5 = byte4 (Tim2) +
                        //byte5 (Tia2)
          //palabra 6 = byte6 (Vacío) +
                        //byte7 (Código de Seguridad = 0x55)

    if ( ((Ecan1Buffer[7][6] & 0xFF00) >> 8) == 0x55 ){

        Kpb2 = Ecan1Buffer[7][3] & 0x00FF;         //byte0 = Kpb2
        Kpm2 = (Ecan1Buffer[7][3] & 0xFF00) >> 8; //byte1 = Kpm2
        Kpa2 = Ecan1Buffer[7][4] & 0x00FF;         //byte2 = Kpa2
        Tib2 = (Ecan1Buffer[7][4] & 0xFF00) >> 8; //byte3 = Tib2
        Tim2 = Ecan1Buffer[7][5] & 0x00FF;         //byte4 = Tim2
        Tia2 = (Ecan1Buffer[7][5] & 0xFF00) >> 8; //byte5 = Tia2

    }//fin del if

    break;

case 0x08: //Interrupción por buffer 8
```

```
            //Ha llegado el mensaje CONF4 --> SID = 0x7F8

            //CONF4 palabra 3 = byte0 (Kpb3) +
                              //byte1 (Kpm3)
                  //palabra 4 = byte2 (Kpa3) +
                              //byte3 (Tib3)
                  //palabra 5 = byte4 (Tim3) +
                              //byte5 (Tia3)
                  //palabra 6 = byte6 (Vacío) +
                              //byte7 (Código de Seguridad = 0x55)

            if ( ((Ecan1Buffer[8][6] & 0xFF00) >> 8) == 0x55 ){

                Kpb3 = Ecan1Buffer[8][3] & 0x00FF;         //byte0 = Kpb3
                Kpm3 = (Ecan1Buffer[8][3] & 0xFF00) >> 8; //byte1 = Kpm3
                Kpa3 = Ecan1Buffer[8][4] & 0x00FF;         //byte2 = Kpa3
                Tib3 = (Ecan1Buffer[8][4] & 0xFF00) >> 8; //byte3 = Tib3
                Tim3 = Ecan1Buffer[8][5] & 0x00FF;         //byte4 = Tim3
                Tia3 = (Ecan1Buffer[8][5] & 0xFF00) >> 8; //byte5 = Tia3

            }//fin del if

            break;

        case 0x41: //Interrupción por ERROR (Error interrupt)
            break;

        default:
            break;

    }//fin del switch

    prepararmsgcan = 1; //para volver a componer los mensajes de STATUS.

    IFS0bits.DMA0IF = 0; //Borrar flag de la interrupción de la DMA canal 0

}//Fin de la RSI del DMA0

//------------------------------------------------------
//     Fin de la RSI del Canal 0 de la DMA (RX)
//------------------------------------------------------




//------------------------------------------------------
//     Inicio de la RSI del Canal 1 de la DMA (TX)
//------------------------------------------------------

void __attribute__((__interrupt__)) _DMA1Interrupt(void){

    //Es necesario tener una interrupción para el envío????

    IFS0bits.DMA1IF = 0; //Borrar flag de la interrupción de la DMA canal 1

}//Fin de la RSI del DMA0

//------------------------------------------------------
//     Fin de la RSI del Canal 1 de la DMA (TX)
//------------------------------------------------------









//*********************************
//     DEFINICIÓN DE FUNCIONES
//*********************************

//------------------------------------------------------
//     Inicio de la funcion "inicializacion_T1"
//------------------------------------------------------

void inicializacion_T1(void){

    //Inicializacion del Timer 1: Timer para generar el periodo de analisis del sistema
```

```
    T1CONbits.TON = 0; //Timer1 apagado
    T1CONbits.TCS = 0; //Seleccionamos como fuente del timer el reloj interno
    T1CONbits.TGATE = 0; //Desactivar modo Gated Timer.
    T1CONbits.TCKPS = 0b11; //Preescaler a 1:256
    TMR1 = 0x0000; //Inicializar el contador del timer a 0
    PR1 = 0xFFFF; //Periodo del timer.
                  //Realmente no hacen falta porque por defecto tras el reset se carga el valor FFFF
                  //Esto generará una interrupción cada PR1+1 = 65536 ciclos del prescaler, que
                  //con el prescaler puesto a 1:256 equivaldría a 0,4194 seg.

    //Nivel de prioridad de interrupción
    IFS0bits.T1IF = 0; //Borrar flag de la interrupción del Timer1
    IEC0bits.T1IE = 1; //Habilitamos las interrupciones provenientes del Timer1

}

//------------------------------------------------------
//     Fin de la funcion "inicializacion_T1"
//------------------------------------------------------




//------------------------------------------------------
//     Inicio de la funcion "inicializacion_T3"
//------------------------------------------------------

void inicializacion_T3(void){

    //Inicialización del Timer 3: Timer para generar el tiempo base del Input Capture

    T3CONbits.TON = 0; //Timer3 apagado
    T3CONbits.TCS = 0; //Seleccionamos como fuente del timer el reloj interno
    T3CONbits.TGATE = 0; //Desactivar modo Gated Timer.
    T3CONbits.TCKPS = 0b10; //Preescaler a 1:64
    TMR3 = 0x0000; //Inicializar el contador del timer a 0
    PR3 = 0xFFFF; //Periodo del timer.
                  //Realmente no hacen falta porque por defecto tras el reset se carga el valor FFFF
                  //Esto generará una interrupción cada PR1+1 = 65536 ciclos del prescaler, que
                  //con el prescaler puesto a 1:64 equivaldría a 0,104 seg.

    //Nivel de prioridad de interrupción
    IFS0bits.T3IF = 0; //Borrar flag de la interrupción del Timer3
    IEC0bits.T3IE = 1; //Habilitamos las interrupciones provenientes del Timer3

}

//------------------------------------------------------
//     Fin de la funcion "inicializacion_T3"
//------------------------------------------------------




//------------------------------------------------------------------
//     Inicio de la funcion "inicializacion_IC1_IC2_IC3_IC4"
//------------------------------------------------------------------

void inicializacion_IC1_IC2_IC3_IC4(void){

 //Inicialización del Input Capture Canal 1. Usado para el cálculo de la velocidad.

    IC1CONbits.ICM = 0b000; //Deshabilitamos el módulo IC1
    IC1CONbits.ICTMR = 0; //Seleccionamos el Timer3 como tiempo base para el IC1
    IC1CONbits.ICI = 0b00; //Generar interrupción en cada evento de captura
    IC1CONbits.ICM = 0b010; //Habilitar modulo IC1 y generar un evento de captura cada flanco
                            //de bajada de la señal conectada al pin IC1.
                            //NOTA: Se hace cada flanco de bajada porque la salida de los sensores
                            //de efecto Hall, es colector abierto y al poner una resistencia de
                            //pull-up en el circuito de captura, la señal se ve "negada".
    //Nivel de prioridad de interrupcion
    IFS0bits.IC1IF = 0; //Borrar flag de la interrupción IC1
    IEC0bits.IC1IE = 1; //Habilitar interrupciones del modulo IC1

 //Inicialización del Input Capture Canal 2. Usado para el cálculo de la velocidad.

    IC2CONbits.ICM = 0b000; //Deshabilitamos el módulo IC2
    IC2CONbits.ICTMR = 0; //Seleccionamos el Timer3 como tiempo base para el IC2
    IC2CONbits.ICI = 0b00; //Generar interrupción en cada evento de captura
```

```
    IC2CONbits.ICM = 0b010; //Habilitar modulo IC2 y generar un evento de captura cada flanco
                            //de bajada de la señal conectada al pin IC2.
                            //NOTA: Se hace cada flanco de bajada porque la salida de los sensores
                            //de efecto Hall, es colector abierto y al poner una resistencia de
                            //pull-up en el circuito de captura, la señal se ve "negada".
    //Nivel de prioridad de interrupcion
    IFS0bits.IC2IF = 0; //Borrar flag de la interrupción IC2
    IEC0bits.IC2IE = 1; //Habilitar interrupciones del modulo IC2

 //Inicialización del Input Capture Canal 3. Usado para el cálculo de la velocidad.

    IC3CONbits.ICM = 0b000; //Deshabilitamos el módulo IC3
    IC3CONbits.ICTMR = 0; //Seleccionamos el Timer3 como tiempo base para el IC3
    IC3CONbits.ICI = 0b00; //Generar interrupción en cada evento de captura
    IC3CONbits.ICM = 0b010; //Habilitar modulo IC3 y generar un evento de captura cada flanco
                            //de bajada de la señal conectada al pin IC3.
                            //NOTA: Se hace cada flanco de bajada porque la salida de los sensores
                            //de efecto Hall, es colector abierto y al poner una resistencia de
                            //pull-up en el circuito de captura, la señal se ve "negada".
    //Nivel de prioridad de interrupcion
    IFS2bits.IC3IF = 0; //Borrar flag de la interrupción IC3
    IEC2bits.IC3IE = 1; //Habilitar interrupciones del modulo IC3

 //Inicialización del Input Capture Canal 4. Usado para el cálculo de la velocidad.

    IC4CONbits.ICM = 0b000; //Deshabilitamos el módulo IC4
    IC4CONbits.ICTMR = 0; //Seleccionamos el Timer3 como tiempo base para el IC4
    IC4CONbits.ICI = 0b00; //Generar interrupción en cada evento de captura
    IC4CONbits.ICM = 0b010; //Habilitar modulo IC4 y generar un evento de captura cada flanco
                            //de bajada de la señal conectada al pin IC4.
                            //NOTA: Se hace cada flanco de bajada porque la salida de los sensores
                            //de efecto Hall, es colector abierto y al poner una resistencia de
                            //pull-up en el circuito de captura, la señal se ve "negada".
    //Nivel de prioridad de interrupcion
    IFS2bits.IC4IF = 0; //Borrar flag de la interrupción IC4
    IEC2bits.IC4IE = 1; //Habilitar interrupciones del modulo IC4

}

//---------------------------------------------------------------------
//     Fin de la funcion "inicializacion_IC1_IC2_IC3_IC4"
//---------------------------------------------------------------------




//-----------------------------------------------------------
//     Inicio de la funcion "inicializacion_T2_PWM"
//-----------------------------------------------------------

void inicializacion_T2_PWM(void){

          //Inicializar la parte del PWM

          OC1CONbits.OCM = 0b0000; //Deshabilitar el modulo Output Compare
          OC1R = 0; //Ciclo de trabajo para el primer pulso del PWM
          OC1RS = 0; //Ciclo de trabajo para el segundo pulso del PWM
          OC1CONbits.OCTSEL = 0; //Seleccionar Timer 2 como base de cuenta para el PWM
          OC1CONbits.OCM = 0b110; // Seleccionar el modo correcto en el Output Compare
                            //(modo PWM sin protección contra fallo)

    //Inicializamos el Timer2 para que sea el encargado de gestionar la frecuencia del PWM.
    //Se configurará en modo cuenta con prescaler 1:256, periodo 65535 y habilitamos las
    //interrupciones. Esto hará que genere una frecuencia de 2,38 Hz ( Fcy=40000000 -->
    //(Fcy/256)/(65535+1) = 2,38 Hz)

          T2CONbits.TON = 0; //Timer apagado
          T2CONbits.TCS = 0; //Seleccionamos como fuente del timer el reloj interno
          T2CONbits.TGATE = 0; //Desactivar modo Gated Timer.
          T2CONbits.TCKPS = 0b10; //Preescaler a 1:64
          TMR2 = 0x0000; //Inicializar el valor de cuenta del timer2 a 0
          PR2 = 0xffff; //Poner el periodo del timer a 65535

          //IPC1bits.T2IF = MIRAR EL TEMA DE INTERRUPCIONES ANIDADAS!!!!!!!
          IFS0bits.T2IF = 0; //Borrar flag de la interrupcion del Timer2
          IEC0bits.T2IE = 1; //Habilita las interrupciones provenientes de Timer2

}

//-----------------------------------------------------------
```

```
//     Fin de la funcion "inicializacion_T2_PWM"
//-------------------------------------------------------------




//-------------------------------------------------------------
//     Inicio de la funcion "inicializacion_ECAN1"
//-------------------------------------------------------------

void inicializacion_ECAN1(void){

    //Inicializacion del módulo ECAN1

    //Reloj del ECAN configurado a 1Mbps
    C1CTRL1bits.REQOP = 4; //Para hacer cambios en los registros que modifican el timing del ECAN
                           //es necesario poner el módulo ECAN en modo de Configuración:
                           //C1CTRL1bits.REQOP = 4.
                           //Realmente, en este momento no sería del todo necesario porque tras
                           //un reset el modulo ECAN ya se encuentra en "modo configuración".
    while (C1CTRL1bits.OPMODE != 4); //esperamos a que el modulo entre en modo de configuracion

    C1CFG2bits.SEG1PH = 0x7; //Phase Segment 1 time = 8TQ
    C1CFG2bits.SEG2PHTS = 0x1; //Activamos el bit de programación del Phase Segment 2
    C1CFG2bits.SEG2PH = 0x5; //Phase Segment 2 time = 6TQ
    C1CFG2bits.PRSEG = 0x4; //Propagation Segment time = 5TQ
    C1CFG2bits.SAM = 0x1; //Muestrear 3 veces el dato en el punto de muestreo
    C1CFG1bits.SJW = 0x3; //Syncronization Jump Width = 4TQ
    C1CFG1bits.BRP = 0x0; //Baud Rate del Fcan = 1Mbps


    C1CTRL1bits.WIN = 0; //WIN = 0 para poder modificar los registros C1FCTRL

    C1FCTRLbits.DMABS = 0b011; //Configuramos ECAN1 para utilizar 12 buffers en la DMA RAM


    //Parte de transmisión - El Buffers 0, 1, 2 y 3 son de transmisión.
    //-----------------------------------------------------------------
    //NOTA: Hay que tener en cuenta que aunque sean de transmisión, estos mensajes son respuesta
    //a una petición remota y así lo debemos configurar (hay que activar un filtro de aceptación).
    //Este caso especial, es el único caso en el que un filtro de aceptación (usados para mensajes
    //que se reciben) apunta a un buffer configurado como transmisión!

    C1CTRL1bits.WIN = 0; //WIN = 0 para poder modificar los registros C1TRmnCON

    C1TR01CONbits.TXEN0 = 0x1; //Buffer 0 configurado como transmisión
    C1TR01CONbits.TXEN1 = 0x1; //Buffer 1 configurado como transmisión
    C1TR23CONbits.TXEN2 = 0x1; //Buffer 2 configurado como transmisión
    C1TR23CONbits.TXEN3 = 0x1; //Buffer 3 configurado como transmisión
    C1TR01CONbits.TX0PRI = 0x0; //Seleccionamos prioridad = Lowest Message Priority
    C1TR01CONbits.TX1PRI = 0x0; //Seleccionamos prioridad = Lowest Message Priority
    C1TR23CONbits.TX2PRI = 0x0; //Seleccionamos prioridad = Lowest Message Priority
    C1TR23CONbits.TX3PRI = 0x0; //Seleccionamos prioridad = Lowest Message Priority
    C1TR01CONbits.RTREN0 = 0x1; //Activamos la respuesta automática a una petición remota
    C1TR01CONbits.RTREN1 = 0x1; //Activamos la respuesta automática a una petición remota
    C1TR23CONbits.RTREN2 = 0x1; //Activamos la respuesta automática a una petición remota
    C1TR23CONbits.RTREN3 = 0x1; //Activamos la respuesta automática a una petición remota

    //Ahora configuramos las máscaras y los filtros de aceptación
    //No vamos a usar máscaras pero no existe la opción de deshabilitarlas, tras el reset los
    //valores por defecto hacen que se use la máscara 0 para todos los filtros de aceptación:
    //C1FMSKSEL1 y C1FMSKSEL2 son 0x0000 tras reset.
    //Lo que vamos a hacer es configurar la mascara 0 para que "no enmascare" el SID del mensaje
    //que recibimos. Para conseguirlo, hay que poner el campo SID de la mascara todo a "1"

    C1CTRL1bits.WIN = 1; //WIN = 1 para poder modificar lo registros de las máscaras y filtros.

    C1FMSKSEL1 = 0x0000; //Realmente no hace falta porque su valor tras reset es 0x0000
                         //Este registro hace que los filtros de aceptación del 0 al 7 usen
                         //la máscara 0.
    C1FMSKSEL1 = 0x0000; //Realmente no hace falta porque su valor tras reset es 0x0000
                         //Este registro hace que los filtros de aceptación del 8 al 15 usen
                         //la máscara 0.

    C1RXM0SIDbits.SID= 0b11111111111; //Máscara 0 todo a "unos" para que "no enmascare".
    C1RXM0SIDbits.MIDE= 1; //Con este bit a 1 podremos configurar que el filtro de aceptación
                           //correspondiente, use o no, los identificadores extendidos según
                           //el valor del bit EXIDE del registro C1RXFnSID (registro de cada filtro)

    //NOTA: Nosotros no usaremos identificadores extendidos. Nuestros mensajes por el BUSCAN solo
```

```
    //usarán identificadores estándar (SID=11bits)

    //Continuamos ahora con los filtros de aceptación.
    //Asignamos los filtros de aceptación 0, 1, 2 y 3 para los buffers 0, 1, 2 y 3 respectivamente

    C1RXF0SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF1SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF2SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF3SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF0SIDbits.SID = 0b11111110000; // Identificador estandar = 0x7F0 = mensaje STATUS0
    C1RXF0SIDbits.SID = 0b11111110001; // Identificador estandar = 0x7F1 = mensaje STATUS1
    C1RXF0SIDbits.SID = 0b11111110010; // Identificador estandar = 0x7F2 = mensaje STATUS2
    C1RXF0SIDbits.SID = 0b11111110011; // Identificador estandar = 0x7F3 = mensaje STATUS3
    C1BUFPNT1bits.F0BP = 0x0; //El filtro de aceptación 0 apunta al Buffer 0.
    C1BUFPNT1bits.F1BP = 0x1; //El filtro de aceptación 1 apunta al Buffer 1.
    C1BUFPNT1bits.F2BP = 0x2; //El filtro de aceptación 2 apunta al Buffer 2.
    C1BUFPNT1bits.F3BP = 0x3; //El filtro de aceptación 3 apunta al Buffer 3.
    C1FEN1bits.FLTEN0 = 1; //Habilitamos el filtro de aceptación 0.
    C1FEN1bits.FLTEN1 = 1; //Habilitamos el filtro de aceptación 1.
    C1FEN1bits.FLTEN2 = 1; //Habilitamos el filtro de aceptación 2.
    C1FEN1bits.FLTEN3 = 1; //Habilitamos el filtro de aceptación 3.


    //Parte de recepción - Los buffers 4, 5, 6, 7 y 8 son para recepción
    //----------------------------------------------------------------------

    C1CTRL1bits.WIN = 0; //WIN = 0 para poder modificar los registros C1TRmnCON

    C1TR45CONbits.TXEN4 = 0x0; //Buffer 4 configurado como recepción
    C1TR45CONbits.TXEN5 = 0x0; //Buffer 5 configurado como recepción
    C1TR67CONbits.TXEN6 = 0x0; //Buffer 6 configurado como recepción
    C1TR67CONbits.TXEN7 = 0x0; //Buffer 7 configurado como recepción
    //NOTA: para el buffer 8 no hace falta indicar que es de recepción, porque a partir del
    //buffer 8 y hasta el buffer 31 todos son solo de recepción.

    //Filtros y máscaras. Nota: La parte de la máscara ya quedó correctamente configurada en la
    //parte de la transmisión, algunas líneas de código más arriba.
    //Asignamos los filtros de aceptación 4, 5, 6, 7 y 8 para los buffers 4, 5, 6, 7 y 8
    //respectivamente.

    C1CTRL1bits.WIN = 1; //WIN = 1 para poder modificar lo registros de los filtros de aceptación

    C1RXF4SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF5SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF6SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF7SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF8SIDbits.EXIDE = 0; //Solo Identificadores Estandar en los mensajes recibidos.
    C1RXF4SIDbits.SID = 0x7F4; // Identificador estandar = 0x7F4 = mensaje CONF0
    C1RXF5SIDbits.SID = 0x7F5; // Identificador estandar = 0x7F5 = mensaje CONF1
    C1RXF6SIDbits.SID = 0x7F6; // Identificador estandar = 0x7F6 = mensaje CONF2
    C1RXF7SIDbits.SID = 0x7F7; // Identificador estandar = 0x7F7 = mensaje CONF3
    C1RXF8SIDbits.SID = 0x7F8; // Identificador estandar = 0x7F8 = mensaje CONF4
    C1BUFPNT2bits.F4BP = 0x4; //El filtro de aceptación 4 apunta al Buffer 4.
    C1BUFPNT2bits.F5BP = 0x5; //El filtro de aceptación 5 apunta al Buffer 5.
    C1BUFPNT2bits.F6BP = 0x6; //El filtro de aceptación 6 apunta al Buffer 6.
    C1BUFPNT2bits.F7BP = 0x7; //El filtro de aceptación 7 apunta al Buffer 7.
    C1BUFPNT3bits.F8BP = 0x8; //El filtro de aceptación 8 apunta al Buffer 8.
    C1FEN1bits.FLTEN4 = 1; //Habilitamos el filtro de aceptación 4.
    C1FEN1bits.FLTEN5 = 1; //Habilitamos el filtro de aceptación 5.
    C1FEN1bits.FLTEN6 = 1; //Habilitamos el filtro de aceptación 6.
    C1FEN1bits.FLTEN7 = 1; //Habilitamos el filtro de aceptación 7.
    C1FEN1bits.FLTEN8 = 1; //Habilitamos el filtro de aceptación 8.


    C1CTRL1bits.REQOP = 2; //Habilitamos el modo Loopback para hacer pruebas.
                           //El modo Loopback hace que lo que enviemos con el ECAN también
                           //lo recibamos y así permitirnos testear el sistema.
    //while (C1CTRL1bits.OPMODE != 2); //esperamos a que el modulo entre en modo Loopback

    C1CTRL1bits.WIN = 0; //WIN = 0 para poder modificar los registros referentes a interrupciones

         //CREO QUE ESTAS INTERRUPCIONES NO SON NECESARIAS. SOLO USARÉ LAS DE LA DMA
    //C1INTEbits.RBIE=1; //Interrupción del modulo ECAN para RX
    //C1INTEbits.TBIE=1; //Interrupción del modulo ECAN para TX

}

//------------------------------------------------------------
//     Fin de la funcion "inicializacion_ECAN1"
//------------------------------------------------------------
```

```
//------------------------------------------------------------
//     Inicio de la funcion "inicializacion_DMA"
//------------------------------------------------------------

void inicializacion_DMA(void){

    //Inicialización del módulo DMA-Controler

    //Canal DMA 0 - Usado para la recepción de datos desde ECAN1 hasta la DMA RAM
    //-------------------------------------------------------------------------

    DMA0REQ = 34; //Asociamos el canal 0 a los eventos de RX del ECAN1

    DMA0CONbits.SIZE = 0x0; //El tamaño de la transferencia de datos se configura a 1 palabra.
    DMA0CONbits.DIR = 0x0; // Dirección de transferencia: Desde Periferico a la DMA RAM

    DMA0CONbits.MODE = 0x0; //Modo de funcionamiento: Continuos Mode, Ping-Pong Disable
    DMA0CONbits.AMODE = 0x2; //Modo de direccionamiento: Peripheral Indirect

    DMA0PAD = (volatile unsigned int) &C1RXD; //Dirección del Periferico para el modo de
                                              //direccionamiento Peripheral Indirect.

    DMA0CNT = 7; //Configuramos n° de transferencias de la DMA para completar un bloque de
                 //transferencia. En el caso del ECAN, un mensaje se compone de 8 palabras,
                 //así que hay que escribir el valor 7 en el registro.

    DMA0STA = __builtin_dmaoffset(Ecan1Buffer); //El puntero del offset del buffer primario del
                                                //módulo DMA canal 0, apunta a los buffers que
                                                //hemos reservado en la memoria DMA RAM.

    IEC0bits.DMA0IE = 1; //Habilitamos las interrupciones del canal 0 de la DMA
    DMA0CONbits.CHEN = 1; //Habilitamos el canal 0 de la DMA.

    //Canal DMA 1 - Usado para la transmisión de datos desde DMA RAM hasta ECAN1
    //-------------------------------------------------------------------------

    DMA1REQ = 70; //Asociamos el canal 1 a los eventos de TX del ECAN1

    DMA1CONbits.SIZE = 0x0; //El tamaño de la transferencia de datos se configura a 1 palabra.
    DMA1CONbits.DIR = 0x1; // Dirección de transferencia: Desde DMA RAM a Periferico

    DMA1CONbits.MODE = 0x0; //Modo de funcionamiento: Continuos Mode, Ping-Pong Disable
    DMA1CONbits.AMODE = 0x2; //Modo de direccionamiento: Peripheral Indirect

    DMA1PAD = (volatile unsigned int) &C1TXD; //Dirección del Periferico para el modo de
                                              //direccionamiento Peripheral Indirect.

    DMA1CNT = 7; //Configuramos n° de transferencias de la DMA para completar un bloque de
                 //transferencia. En el caso del ECAN, un mensaje se compone de 8 palabras,
                 //así que hay que escribir el valor 7 en el registro.

    DMA1STA = __builtin_dmaoffset(Ecan1Buffer); //El puntero del offset del buffer primario del
                                                //módulo DMA canal 1, apunta a los buffers que
                                                //hemos reservado en la memoria DMA RAM.

    IEC0bits.DMA1IE = 1; //Habilitamos las interrupciones del canal 1 de la DMA
    DMA1CONbits.CHEN = 1; //Habilitamos el canal 1 de la DMA.

}

//------------------------------------------------------------
//     Fin de la funcion "inicializacion_DMA"
//------------------------------------------------------------




//------------------------------------------------------------
//     Inicio de la funcion "inicializacion_puertos"
//------------------------------------------------------------

void inicializacion_puertos(void){

    //Según el manual, los pines que no se usen deben ser comfigurados como salidas y llevados
    //a estado de cero lógico.
```

```
    TRISA = 0x0000; //Puerto A como salida
    PORTA = 0x0000; //Inicializar el puerto A a 0

    TRISB = 0x0000;
    PORTB = 0x0000;

    TRISC = 0x0000;
    PORTC = 0x0000;

    TRISD = 0x0000;
    PORTD = 0x0000;

    TRISE = 0x0000;
    PORTE = 0x0000;

    TRISF = 0x0000;
    PORTF = 0x0000;

    TRISG = 0x0000;
    PORTG = 0x0000;

    //Los pines del Input Capture son "user settable" eso significa que aunque el modulo del
    //Input Capture esté activado, nosotros podemos configurar el pin como salida y escribir
    // en él para simular un evento en el Input Capture. Por todo esto, para estos pines en los
    //que el modulo no toma control absoluto de ellos, es necesario configurarlos correctamente.
    //Pasaremos a establecer los pines IC1(RD8), IC2(RD9), IC3(RD10) y IC4(RD11) como entrada.

    TRISD = 0b0000111100000000;

    //Los pines RG0, RG14, RG13 y RG12 son los pines que controlan el encendido de los leds del
    //control remoto y los tengo que configurar como salidas en drenador abierto
    //Además de esto, los dejaremos apagados, poniendo los puertos a 0 para que la salida en
    //colector abierto se conecte a GND y no se ponga en alta impedancia.

    ODCG = 0b0111000000000001;
    LATGbits.LATG13 = 0;  //LEDTCSOFF
    LATGbits.LATG12 = 0;  //LEDMAPA1
    LATGbits.LATG14 = 0;  //LEDMAPA2
    LATGbits.LATG0 = 0;   //LEDMAPA3

    //Los pines RE3 y RE4 leen el estado del sistema de control de tracción. Los tengo que
    //configurar como entrada. Como son pines analógicos, también voy a escribir en el registro
    //del ADC1 para establecerlos como digitales, aunque esto, si el módulo no está habilitado
    //no se si haría falta.

    TRISE = 0b0000000000011000;
    AD1PCFGH = 0b0001100000000000;

    //Los pines de programación: RGEC1(RB6) y RGED1(RB7) aunque no los voy a usar, sí están
    //conectados, por lo que los voy a configurar como vienen por defecto: entrada digital.

    TRISB = 0b0000000011000000;

}

//------------------------------------------------------------
//     Fin de la funcion "inicializacion_puertos"
//------------------------------------------------------------




//------------------------------------------------------------
//     Inicio de la funcion "composicion_msgcan"
//------------------------------------------------------------

void composicion_msgcan(void){

    unsigned char msgtemp1 = 0;
    unsigned char msgtemp2 = 0;

    // Código del BUS CAN. Composición de las variables a enviar referentes al BUS CAN

    //Preparamos los mensajes de estatus: STATUS0, STATUS1, STATUS2 y STATUS3
    //Serán todos Tramas Estándar.
    //Los SID serán:
    //STATUS0 = 0x7F0 = 0b111 1111 0000 (se guardará en el Buffer0 = Ecan1Buffer[0])
    //STATUS1 = 0x7F1 = 0b111 1111 0001 (se guardará en el Buffer1 = Ecan1Buffer[1])
    //STATUS2 = 0x7F2 = 0b111 1111 0010 (se guardará en el Buffer2 = Ecan1Buffer[2])
    //STATUS3 = 0x7F3 = 0b111 1111 0011 (se guardará en el Buffer3 = Ecan1Buffer[3])
```

```
//Un mensaje se compone de 8 palabras de 16bits: (los bits "x" son bits que no se usan)
    //Palabra 0 = xxx(3bits) + SID(11bits) + SRR(1bit) + IDE(1bit)
    //Palabra 1 = xxxx(4bits) + EID(12bits) - Pero no usaremos identificadores extendidos
    //Palabra 2 = EID (6bits) + RTR(1bit) + RB1(1bit)+ xxx(3bits) + RB0(1bit) + DLC(4bits)
    //Palabra 3 = DATA byte 1 + DATA byte 0
    //Palabra 4 = DATA byte 3 + DATA byte 2
    //Palabra 5 = DATA byte 5 + DATA byte 4
    //Palabra 6 = DATA byte 7 + DATA byte 6
    //Palabra 7 = xxx(3bits) + FIHIT(5bits) + xxxxxxxx(8bits)
                //FIHIT: Filtro que ha originado el guardado en la DMA RAM.
                //Solo para recepciones

//-------------------------STATUS0 se compondrá de:-----------------------------
//Palabra 0:
//IDE = 0 (Standart Data Frame)
//SRR = 0 (Mensaje normal. No es una peticion de respuesta remota)
//SID = 0b11111110000 = 0x7F0
//Palabra 0 = 0bxxx1 1111 1100 0000

Ecan1Buffer[0][0]= 0b0001111111000000;

//Palabra 1:
//EID = no lo usaremos
//Palabra 1 = 0b0000 0000 0000 0000

Ecan1Buffer[0][1]= 0x0000;

//Palabra 2:
//EID = no lo usaremos
//RTR = 0 (no es una trama de peticion de respuesta remota)
//RB0 = RB1 = 0 (hay que ponerlo a siempre a 0)
//DLC = 0b1000 (tamaño del dato que mandamos = 8 bytes)
//Palabra 2 = 0b0000 0000 0000 1000

Ecan1Buffer[0][2]= 0b0000000000001000;

//Palabra 3: CIRCUNFERENCIA DE RUEDA DELANTERA
//La palabra 3 se compone de los bytes 0 y 1 del dato a enviar.
//En el byte 0 guardaremos la parte decimal de la longitud de la circunferencia (unsigned char)
//En el byte 1 guardaremos la parte entera de la longitud de la circunferencia (unsigned char)

msgtemp1 = (char)circunferenciadelantera;
msgtemp2 = (char)(((circunferenciadelantera+0.001)-(char)circunferenciadelantera)*100);
Ecan1Buffer[0][3]= (msgtemp1 << 8) | (msgtemp2);

//Palabra 4: CIRCUNFERENCIA DE RUEDA TRASERA
//La palabra 4 se compone de los bytes 2 y 3 del dato a enviar.
//En el byte 2 guardaremos la parte decimal de la longitud de la circunferencia (unsigned char)
//En el byte 3 guardaremos la parte entera de la longitud de la circunferencia (unsigned char)

msgtemp1 = (char)circunferenciatrasera;
msgtemp2 = (char)(((circunferenciatrasera+0.001)-(char)circunferenciatrasera)*100);
Ecan1Buffer[0][4]= (msgtemp1 << 8) | (msgtemp2);

//Palabra 5 y 6: %ERROR PERMITIDO Y N°DIENTES
//La palabra 5 se compone de los bytes 4 y 5 del dato a enviar.
//La palabra 6 se compone de los bytes 6 y 7 del dato a enviar
//En el byte 4 guardaremos el %error permitido a baja velocidad (unsigned char)
//En el byte 5 guardaremos el %error permitido a media velocidad (unsigned char)
//En el byte 6 guardaremos el %error permitido a alta velocidad (unsigned char)
//En el byte 7 guardaremos el n°dientes del disco dentado (unsigned char)

Ecan1Buffer[0][5]= (errormediavelocidad << 8) | (errorbajavelocidad);
Ecan1Buffer[0][6]= (dientes << 8) | (erroraltavelocidad);


//-------------------------STATUS1 se compondrá de:-----------------------------
//Palabra 0, 1 y 2 son iguales que en el caso de STATUS0, salvo que el SID será = 0x7F1

Ecan1Buffer[1][0]= 0b0001111111000100;
Ecan1Buffer[1][1]= 0x0000;
Ecan1Buffer[1][2]= 0b0000000000001000;

//Palabra 3, 4, 5 y 6: CONSTANTES DEL REGULADOR MAPA1 Y UMBRALES DE VELOCIDAD
//La palabra 3 se compone de los bytes 0 y 1 del dato a enviar.
//La palabra 4 se compone de los bytes 2 y 3 del dato a enviar.
//La palabra 5 se compone de los bytes 4 y 5 del dato a enviar.
//La palabra 6 se compone de los bytes 6 y 7 del dato a enviar
//En el byte 0 guardaremos la Constante Proporcional a baja velocidad del MAPA1 (unsigned char)
```

```
//En el byte 1 guardaremos la Constante Proporcional a media velocidad del MAPA1 (unsigned char)
//En el byte 2 guardaremos la Constante Proporcional a alta velocidad del MAPA1 (unsigned char)
//En el byte 3 guardaremos el Tiempo Integral a baja velocidad del MAPA1 (unsigned char)
//En el byte 4 guardaremos el Tiempo Integral a media velocidad del MAPA1 (unsigned char)
//En el byte 5 guardaremos el Tiempo Integral a alta velocidad del MAPA1 (unsigned char)
//En el byte 6 guardaremos el valor del umbral a baja velocidad (unsigned char)
//En el byte 7 guardaremos el valor del umbral a alta velocidad (unsigned char)

//IMPORTANTE: los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados
//multiplicados por 10. En el programa principal, estos valores tambien se almacenan
//multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión
//alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 --> Kpb1=0.1
                                                   //Si guardamos el valor 123 --> Kpb1=12.3
//Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores
//tomados por las constanstes proporcionales y los tiempos integrales del regulador
//comprenderán el rango de 0 <= Kp/Ti <= 25.5 y siempre con incrementos de 0.1

Ecan1Buffer[1][3]= (Kpm1 << 8) | (Kpb1);
Ecan1Buffer[1][4]= (Tib1 << 8) | (Kpa1);
Ecan1Buffer[1][5]= (Tia1 << 8) | (Tim1);
Ecan1Buffer[1][6]= (ALTAVELOCIDAD << 8) | (BAJAVELOCIDAD);


//-------------------------STATUS2 se compondrá de:-----------------------------
//Palabra 0, 1 y 2 son iguales que en el caso de STATUS0, salvo que el SID será = 0x7F2

Ecan1Buffer[2][0]= 0b0001111111001000;
Ecan1Buffer[2][1]= 0x0000;
Ecan1Buffer[2][2]= 0b0000000000001000;

//Palabra 3, 4, 5 y 6: CONSTANTES DEL REGULADOR MAPA2
//La palabra 3 se compone de los bytes 0 y 1 del dato a enviar.
//La palabra 4 se compone de los bytes 2 y 3 del dato a enviar.
//La palabra 5 se compone de los bytes 4 y 5 del dato a enviar.
//La palabra 6 se compone de los bytes 6 y 7 del dato a enviar
//En el byte 0 guardaremos la Constante Proporcional a baja velocidad del MAPA2 (unsigned char)
//En el byte 1 guardaremos la Constante Proporcional a media velocidad del MAPA2 (unsigned char)
//En el byte 2 guardaremos la Constante Proporcional a alta velocidad del MAPA2 (unsigned char)
//En el byte 3 guardaremos el Tiempo Integral a baja velocidad del MAPA2 (unsigned char)
//En el byte 4 guardaremos el Tiempo Integral a media velocidad del MAPA2 (unsigned char)
//En el byte 5 guardaremos el Tiempo Integral a alta velocidad del MAPA2 (unsigned char)
//En el byte 6 estará vacio, lo rellenaremos con 0x00
//En el byte 7 estará vacio, lo rellenaremos con 0x00

//IMPORTANTE: los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados
//multiplicados por 10. En el programa principal, estos valores tambien se almacenan
//multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión
//alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 --> Kpb1=0.1
                                                   //Si guardamos el valor 123 --> Kpb1=12.3
//Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores
//tomados por las constanstes proporcionales y los tiempos integrales del regulador
//comprenderán el rango de 0 <= Kp/Ti <= 25.5 y siempre con incrementos de 0.1

Ecan1Buffer[2][3]= (Kpm2 << 8) | (Kpb2);
Ecan1Buffer[2][4]= (Tib2 << 8) | (Kpa2);
Ecan1Buffer[2][5]= (Tia2 << 8) | (Tim2);
Ecan1Buffer[2][6]= 0x0000;


//-------------------------STATUS3 se compondrá de:-----------------------------
//Palabra 0, 1 y 2 son iguales que en el caso de STATUS0, salvo que el SID será = 0x7F3

Ecan1Buffer[3][0]= 0b0001111111001100;
Ecan1Buffer[3][1]= 0x0000;
Ecan1Buffer[3][2]= 0b0000000000001000;

//Palabra 3, 4, 5 y 6: CONSTANTES DEL REGULADOR MAPA3
//La palabra 3 se compone de los bytes 0 y 1 del dato a enviar.
//La palabra 4 se compone de los bytes 2 y 3 del dato a enviar.
//La palabra 5 se compone de los bytes 4 y 5 del dato a enviar.
//La palabra 6 se compone de los bytes 6 y 7 del dato a enviar
//En el byte 0 guardaremos la Constante Proporcional a baja velocidad delMAPA3 (unsigned char)
//En el byte 1 guardaremos la Constante Proporcional a media velocidad delMAPA3 (unsigned char)
//En el byte 2 guardaremos la Constante Proporcional a alta velocidad delMAPA3 (unsigned char)
//En el byte 3 guardaremos el Tiempo Integral a baja velocidad delMAPA3 (unsigned char)
//En el byte 4 guardaremos el Tiempo Integral a media velocidad delMAPA3 (unsigned char)
//En el byte 5 guardaremos el Tiempo Integral a alta velocidad delMAPA3 (unsigned char)
//En el byte 6 estará vacio, lo rellenaremos con 0x00
//En el byte 7 estará vacio, lo rellenaremos con 0x00
```

```
    //IMPORTANTE: los valores de "Constante Proporcional" y "Tiempo Integral" serán guardados
    //multiplicados por 10. En el programa principal, estos valores tambien se almacenan
    //multiplicados por 10 hasta el momento de su uso, por lo que no tenemos que hacer conversión
    //alguna, pero sí es importante saberlo. Ejemplos: Si guardamos el valor 1 --> Kpb1=0.1
                                                        //Si guardamos el valor 123 --> Kpb1=12.3
    //Como la variable unsigned char solo guarda valores de 0 a 255, el rango de los valores
    //tomados por las constanstes proporcionales y los tiempos integrales del regulador
    //comprenderán el rango de 0 <= Kp/Ti <= 25.5 y siempre con incrementos de 0.1

    Ecan1Buffer[3][3]= (Kpm3 << 8) | (Kpb3);
    Ecan1Buffer[3][4]= (Tib3 << 8) | (Kpa3);
    Ecan1Buffer[3][5]= (Tia3 << 8) | (Tim3);
    Ecan1Buffer[3][6]= 0x0000;




}

//-------------------------------------------------------------
//     Fin de la funcion "composicion_msgcan"
//-------------------------------------------------------------




//-------------------------------------------------------------
//     Inicio de la funcion "calcularvelocidadglobal"
//-------------------------------------------------------------

float calcularvelocidadglobal(float dd1, float di1){

          float velocidad = 20;

          if (dd1>=di1)
                    velocidad = dd1; //velocidad del vehiculo en Km/h
          else
                    velocidad = di1; //velocidad del vehiculo en Km/h

          return (velocidad);

    //¿Por que doy la velocidad más alta de entre las ruedas delanteras? En principio descarto
    //las ruedas traseras porque son las ruedas motrices y podrian tener deslizamiento debido
    //a la posición del acelerador y adherencia. De entre las ruedas delanteras elijo la que
    //vaya más rápido porque al no ser motrices no pueden "sobregirar" y es posible que en una
    //curva la rueda interior quede bloqueada debido a la acción del freno y la fuerza lateral.
    //De este modo optamos por elejir en todo momento a la rueda delantera que marca más
    //velocidad para calcular la velocidad del coche.
}

//-------------------------------------------------------------
//     Fin de la funcion "calcularvelocidadglobal"
//-------------------------------------------------------------




//-------------------------------------------------------------
//     Inicio de la funcion "evaluar"
//-------------------------------------------------------------

float evaluar (float dd1, float di1, float td1, float ti1, float velocidad){

          float deslizamiento=0;
          float deslizamientoderecho=0,deslizamientoizquierdo=0;

    //Aplicamos el umbral para el cálculo del deslizamiento. Para ello, toda velocidad por debajo
    //de 1 km/h, la dejaremos en 1 km/h. Con esto evitaremos división por 0 en la formula del
    //cálculo de deslizamiento y también evitaremos tener altos % de deslizamiento por movernos
    //en magnitudes cercanas a 0.

    if (dd1 < 1){
        dd1 = 1;
    }
    if (di1 < 1){
        di1 = 1;
    }
    if (td1 < 1){
        td1 = 1;
    }
```

```
    if (ti1 < 1){
        ti1 = 1;
    }

    if (dd1 < 1 && di1 < 1 && td1 < 1 && ti1 < 1){ //Como si el coche estuviera parado
        deslizamiento = 0;
    }

    else {

        //Calculamos el deslizamiento derecho
        if (dd1 >= td1){
            deslizamientoderecho = (dd1 - td1) / dd1; //poniendo en el primer término de la resta
                                                      //el mayor de los dos valores de velocidad
                                                      //conseguimos que no salga nunca un valor
                                                      //negativo en el deslizamiento.
        }
        else {
            deslizamientoderecho = (td1 - dd1) / td1;
        }


         //Calculamos el deslizamiento izquierdo
        if (di1 >= ti1){
            deslizamientoizquierdo = (di1 - ti1) / di1;
        }
        else {
            deslizamientoizquierdo = (ti1 - di1) / ti1;
        }

         //Nos quedamos con el mayor deslizamiento detectado
         if (deslizamientoderecho >= deslizamientoizquierdo)
                  deslizamiento = deslizamientoderecho * 100; //multipl. por 100 para obtener en %
         else
                  deslizamiento = deslizamientoizquierdo * 100; //multipl. por 100 para obtener en %

    }//fin del else

         //Filtramos por velocidad y aplicamos el error permitido (que está en %)

         if (velocidad >= ALTAVELOCIDAD){
                  if (deslizamiento <= erroraltavelocidad)
                           deslizamiento=0;
         } //fin del if ALTAVELOCIDAD

         else if (velocidad >= BAJAVELOCIDAD && velocidad < ALTAVELOCIDAD){
                  if (deslizamiento <= errormediavelocidad)
                           deslizamiento=0;
         } //fin del else if MEDIAVELOCIDAD

         else if (velocidad >= 0 && velocidad < BAJAVELOCIDAD){
                  if (deslizamiento <= errorbajavelocidad)
                           deslizamiento=0;
         } //fin del else if BAJAVELOCIDAD

         return(deslizamiento/100); //Devolver el valor de deslizamiento en tanto por uno,
                                    //que sería el error del sistema.

}//fin de la funcion "evaluar".

//-------------------------------------------------------------
//     Fin de la funcion "evaluar"
//-------------------------------------------------------------




//-------------------------------------------------------------
//     Inicio de la funcion "devolverconstantes"
//-------------------------------------------------------------

void devolverconstantes(float velocidad, unsigned char mapa, float *Kp, float *Ti){

    //Con esta función, dejamos cargado el valor correcto de las constantes del regulador
    //que usaremos para generar la acción de control que se calculará en la
    //función "float control(float error, float kp, float ki)"
    //Ademas, normalizamos el valor de las constantes porque recordamos que se guardaban
    //multiplicadas por 10, para poderlas almacenar en variables unsigned char. Esto además
    //nos ahorraba trabajo a la hora de componer el mensaje a enviar por el BUS CAN.
```

```
          if (velocidad >= ALTAVELOCIDAD){
                    if (mapa == MAPA1){
                     *Kp=(float)Kpa1/10;
                     *Ti=(float)Tia1/10;
                    }
                    else if (mapa == MAPA2){
                     *Kp=(float)Kpa2/10;
                     *Ti=(float)Tia2/10;
                    }
                    else if (mapa == MAPA3) {
                     *Kp=(float)Kpa3/10;
                     *Ti=(float)Tia3/10;
                    }
          } /*fin del if ALTAVELOCIDAD*/

          if (velocidad >= BAJAVELOCIDAD && velocidad < ALTAVELOCIDAD){
                    if (mapa == MAPA1){
                     *Kp=(float)Kpm1/10;
                     *Ti=(float)Tim1/10;
                    }
                    else if (mapa == MAPA2){
                     *Kp=(float)Kpm2/10;
                     *Ti=(float)Tim2/10;
                    }
                    else if (mapa == MAPA3){
                     *Kp=(float)Kpm3/10;
                     *Ti=(float)Tim3/10;
                    }
          } /*fin del if MEDIAVELOCIDAD*/

          if (velocidad >= 0 && velocidad < BAJAVELOCIDAD){
                    if (mapa == MAPA1){
                     *Kp=(float)Kpb1/10;
                     *Ti=(float)Tib1/10;
                    }
                    else if(mapa == MAPA2){
                     *Kp=(float)Kpb2/10;
                     *Ti=(float)Tib2/10;
                    }
                    else if (mapa == MAPA3){
                     *Kp=(float)Kpb3/10;
                     *Ti=(float)Tib3/10;
                    }
          } /*fin del if BAJAVELOCIDAD*/

}//fin de la función "devolverconstantes"

//-------------------------------------------------------------
//     Fin de la funcion "devolverconstantes"
//-------------------------------------------------------------




//-------------------------------------------------------------
//     Inicio de la funcion "control"
//-------------------------------------------------------------

float control(float error,float Kp,float Ti, float velocidadglobal){

    //Hay que tener en cuenta, en nuestro regulador PI, que el proceso que tenemos que controlar
    //no necesita de un aporte continuo de la señal de control para que el error en régimen
    //permanente sea 0. También debemos tener en cuenta que el error acumulado no se decrementará
    //porque nunca tendremos una señal de error negativa, así que debemos incluir estas
    //características al algoritmo de control.

    //Lo que vamos a hacer es que cuando el error se haga cero, resetearemos el valor de la
    //acción integral porque no nos hará falta.

    float salida = 0;

    if (manteneraccion != 0 && error==0){
        salida = 0.1;
        manteneraccion = manteneraccion - 1;
    }
    else if (error == 0){
        erroracumulado = 0;
        salida = 0;
    }
    else {
```

```
        salida=(error*Kp)+(erroracumulado/Ti);
        erroracumulado = erroracumulado + error;
        if (velocidadglobal < BAJAVELOCIDAD){
            manteneraccion = 5;
        }
    }

          return (salida);

}//fin de la función "control"

//------------------------------------------------------------
//     Fin de la funcion "control"
//------------------------------------------------------------




//------------------------------------------------------------
//     Inicio de la funcion "actuar"
//------------------------------------------------------------

void actuar(float salida){

    //La función actuar, está diseñada para generar una señal cuadrada de ciclo de trabajo variable
    //proporcional a la señal generada por la función de control.
    //Aplicaremos un algoritmo que por un lado, restrinja el valor máximo de ciclo de trabajo
    //al 30% y por otro limite los incrementos del ciclo de trabajo a saltos del 10%.


          float CT = 0; //Ciclo de trabajo

    if (salida == 0){
        CT = 0;
    }
    else if (salida > 0 && salida <= 0.1){
        CT = 10;
    }
    else if (salida > 0.1 && salida <=0.2){
        CT = 20;
    }
    else if (salida > 0.2 && salida <=0.3){
        CT = 30;
    }
    else {
        CT = 30;
    }

          /*Generar el valor a cargar en OC1RS dependiendo del CT a generar*/
    if (CT == 0){
        OC1RS = 0;
    }
    else {
        OC1RS = (int)(PR2 / (100/CT));
    }

} //fin de la función "actuar"

//------------------------------------------------------------
//     Fin de la funcion "actuar"
//------------------------------------------------------------
```